Software Engineering Institute

FloCon 2005 Proceedings

September 2005

CERT Program

http://www.sei.cmu.edu

Carnegie Mellon

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **SEP 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **FloCon 2005 Proceedings**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University,Software Engineering Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**321**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

NVisionIP: An Animated State Analysis Tool for Visualizing NetFlows

Ratna Bearavolu Kiran Lakkaraju William Yurcik

National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
{ratna,kiran,byurcik}@ncsa.uiuc.edu

***Abstract*—In this paper, we describe a NetFlow visualization tool, NVisionIP, which provides network administrators increased situational awareness of the state of networked devices within an IP address space. It does this by providing three increasingly detailed views of the state of devices in an entire IP address space to subnets to individual machines. Operators may use NVisionIP to transparently view NetFlow traffic without filtering or may selectively filter and interactively query NVisionIP for unique views given experience or relevant clues.**

***Index Terms*— NetFlows, Visualization, Network Security**

I. Introduction

What is the state of devices on your large and complex network? This is a question management commonly poses to network administrators and up to now the answer has been problematic. IDS sensors give binary alarms for signature-matches or anomalous traffic, if no alarms then there is no state information about the devices on the network. Scans test for software vulnerabilities but this is more about predicting posture to future attacks than knowledge of current state. Network device monitoring devices like MRTG[1] and the Flowscan[2] may display traffic levels by service as well as aggregate traffic load levels – while this is certainly useful for managing traffic congestion and detecting high volume events, there are no details about device state and small events are obscured.

While NetFlows provide an excellent source of information concerning the behavior of the network, the sheer magnitude of NetFlow logs often makes it difficult to gain an understanding of that behavior. In this paper we present a tool, NVisionIP [1,3-5,9-11], that uses NetFlows to visually represent activity on an entire IP address space. NVisionIP presents information at three different levels allowing operators to select which level to use.

II. System Architecture

The NVisionIP system architecture is comprised of three modules: Data Retrieval Module, Computation Module and Visualization Module. As shown in Figure 1, the three modules interact using a Mediator object [2]. By using a Mediator object, we avoid direct referencing of a module by other modules, thus providing the flexibility of modifying the modules independently. The Data Retrieval Module reads in the NetFlow files, preprocesses them, and places them in a table structure for the Computation module to use. For every IP address in the input table, the Computation Module calculates various statistics as shown in Figure 2. These statistics are then passed to the Visualization Module that presents information to a user.

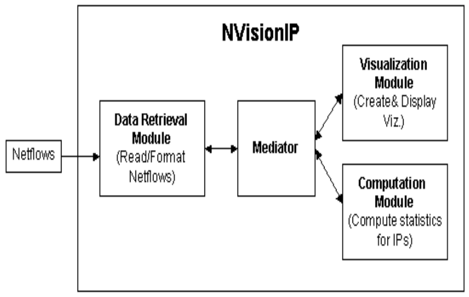

NVisionIP
Netflows
Data Retrieval Module (Read/Format Netflows)
Mediator
Visualization Module (Create& Display Viz.)
Computation Module (Compute statistics for IPs)

Figure 1. NVisionIP System Architecture

- *Number of times IP address part of a flow*
- *Number of times IP address destination of a flow*
- *Number of times IP address source of a flow*
- *Number of ports used by IP address*
- *Number of destination ports used by IP address*
- *Number of source ports used by IP address*
- *Number of protocols used by IP address*
- *Number of bytes transmitted to/from IP address*
- *Number of bytes transmitted to/from IP address*

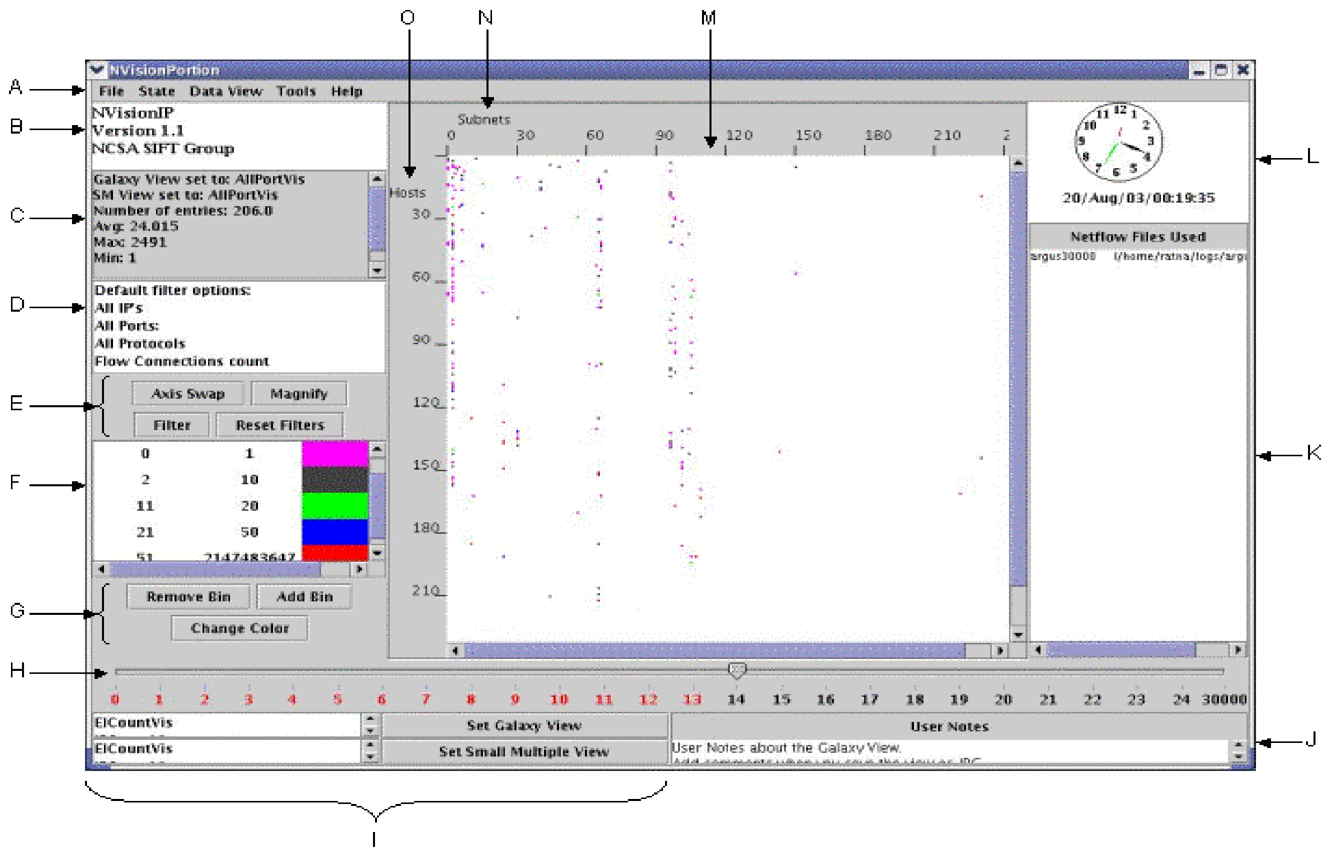
Figure 2. Statistics Derived from NetFlows

III. How to use NVisionIP

NVisionIP can be downloaded here:
<http://security.ncsa.uiuc.edu/distribution/NVisionIPDownLoad.html>

NVisionIP builds on the concept of "overview, browse, drill-down to details-on-demand" championed by Shneiderman [6] and Tufte [8] to support three different views: (1) a Galaxy View (GV) - a high-level view of an entire network, (2) a Small Multiple View (SMV) - a subnet view of traffic from multiple machines within the network, and (3) a Machine View (MV) information about flows into/out of a single machine. Overview plus detail breaks up content into comprehensible pieces while also allowing for simultaneous comparisons of different views which may reveal interrelationships [7].

[1] Multi Router Traffic Grapher <http://mrtg.hdl.com/mrtg.html>
[2] <http://net.doit.wisc.edu/~plonka/FlowScan/>



NVisionIP
Version 1.1
NCSA SIFT Group

Galaxy View set to: AllPortVis
SM View set to: AllPortVis
Number of entries: 206.0
Avg: 24.015
Max: 2491
Min: 1

Default filter options:
All IP's
All Ports:
All Protocols
Flow Connections count

0	1	🟪
2	10	⬛
11	20	🟩
21	50	🟦
51	2147483647	🟥

20/Aug/03/00:19:35

argus30000 /home/ratna/logs/arg

ElCountVis
ElCountVis

User Notes about the Galaxy View.

A.	Menu Bar
B.	NVisionIP version number and other information
C.	Summary of statistics being displayed on the grid
D.	Current filter options being applied to the grid – default are displayed in the picture
E.	Axis Swap – To swap the subnet and hosts axis; Magnify – Apply a magnifier around the cursor area to enhance the grid; Filter – Apply user specified filter to the data; Reset Filter – Remove any filters applied to the data.
F.	Legend mapping a range of numbers to a color
G.	Functionalities to customize the legend (F)
H.	Slider showing different intervals of data, 30000 indicates the total number of netflows the user selected to process. In the image, the user moved the slider bar to interval 14. This implies that (1200*14) netflows have been processed and displayed
I.	Options for the user to select which attribute of the machines the Galaxy view and Small Multiple view should display. In the image, it is set to show number of unique ports used by the machines (See Panel C)
J.	Notes area for the user to make comments about the GV. The notes get saved, when the user saves the GV as an image.
K.	List of netflow files that are loaded into the system
L.	Shows the time stamp of the last netflow processed among the selected intervals.
M.	XY grid that shows a class B network
N.	Subnets axis
O.	Hosts axis

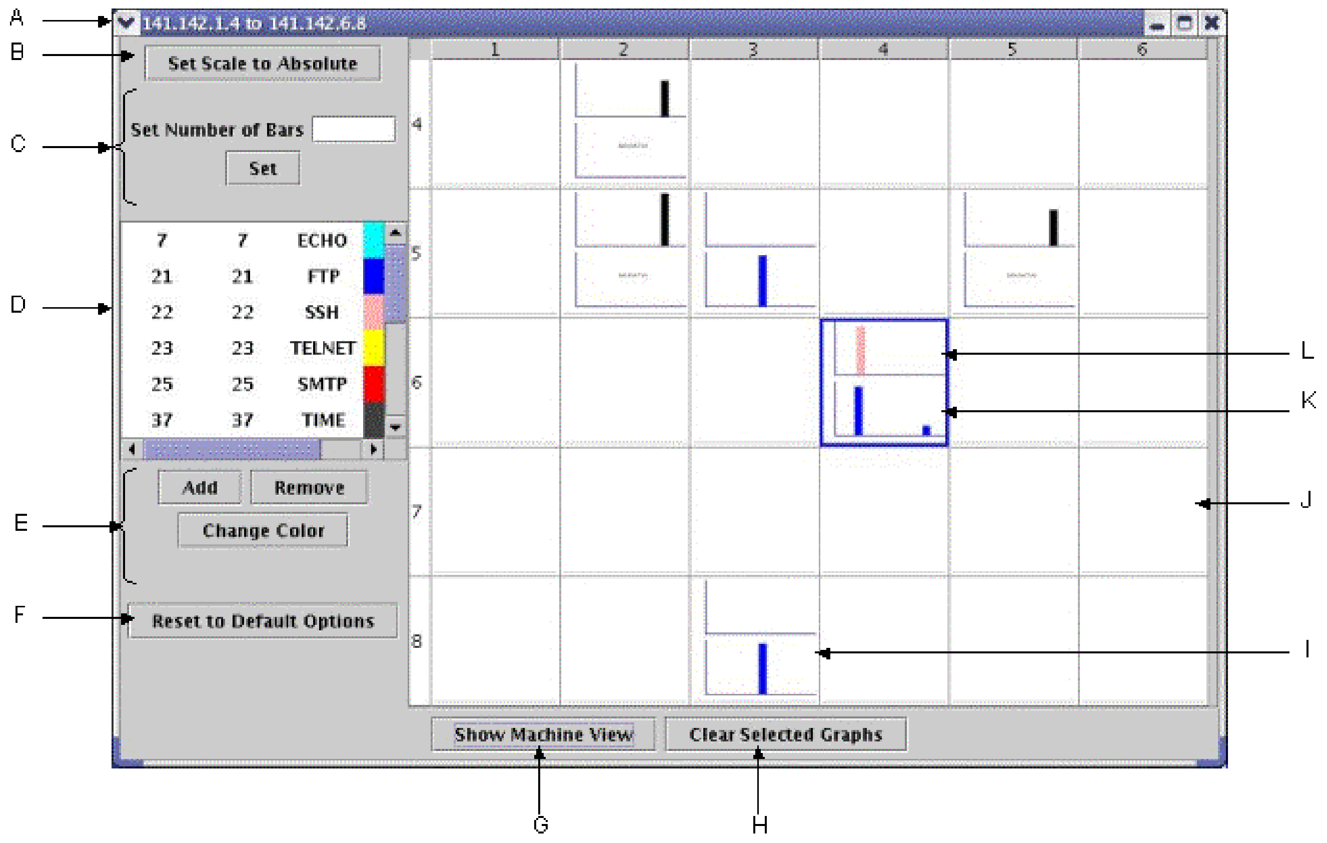
Figure 3. NVisionIP Galaxy View (GV)

Figure 3 shows the GV, a 2D graph of a Class B network where the hosts are on the X-axis and subnets are on the Y-axis—this orientation can be changed by the *Swap Axis Button* (Figure 3:Label E). A single point on the graph represents an IP address on the network being monitored. For example, point (50, 70) on the graph represents the IP address 141.142.50.70. An IP address displays information about any of the statistics from Figure 2 using a color code. The mapping from number to color is provided by the customizable binning legend (Figure 3:Label F). The default statistic configured in the Galaxy view is the number of unique ports used by the host. The *Set Galaxy View Button* (Figure 3: Label I) can change this view.

To process input data, the user selects NetFlow files to be visualized using the menu file pull down. NVisionIP divides input files into intervals of equal numbers of flows (user selects number of intervals). The last number on the slider bar reflects the total number of NetFlows loaded.

When the user moves the bar across the data intervals, NVisionIP provides the option of either viewing the results as a summation (cumulative view) or piece-wise (animated view). For example, if the user chooses a cumulative view and moves the slider (Figure 3:Label H) from interval 0 to 4, then the results that are displayed for all the IPs are the summation of the values (Ports, Protocols, Count, etc) of NetFlows from interval 0 to 4. If the user had selected the animation view and moved the slider from interval 0 to 4, the GV would show a 4 frame animation—each frame representing activity during one time interval. The animation shows how IP device state changes over time, providing a temporal feel for device state on the network.

To learn more about the GV filtering option see [3,5]. GV magnification and storage options are described within the application itself.



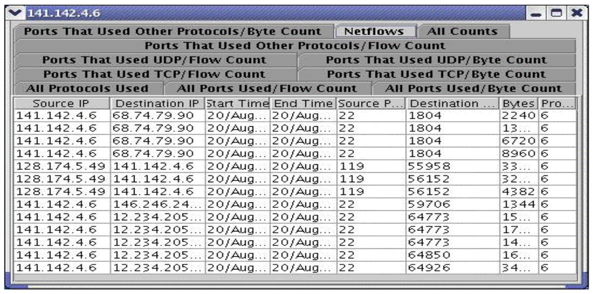
7	7	ECHO	🟦
21	21	FTP	🟦
22	22	SSH	🟥
23	23	TELNET	🟨
25	25	SMTP	🟥
37	37	TIME	⬛

A.	Range of IPs being displayed	G.	Shows the machine view of a selected machine
B.	Sets the scale of the bar charts axis – to absolute or relative (default)	H.	Clears any selected machine (graph)
C.	Sets the number of bars to displayed for every machine	I.	Machine 141.142.3.8 in SMV
D.	Legend mapping port/Protocol number to a color	J.	Machine 141.142.6.7 – An empty panel, as no information is contained by NVisionIP
E.	Functionalities to customize the legend	K.	Lower graph – showing other ports (all ports – special ports (from the legend)) used by 141.142.4.6
F.	Resets the options to default (Scale, Number of bars, Port/Protocol mapping to default color)	L.	Upper graph – showing the special ports used by 141.142.4.6

Figure 4. NVisionIP Small Multiple View (SMV)

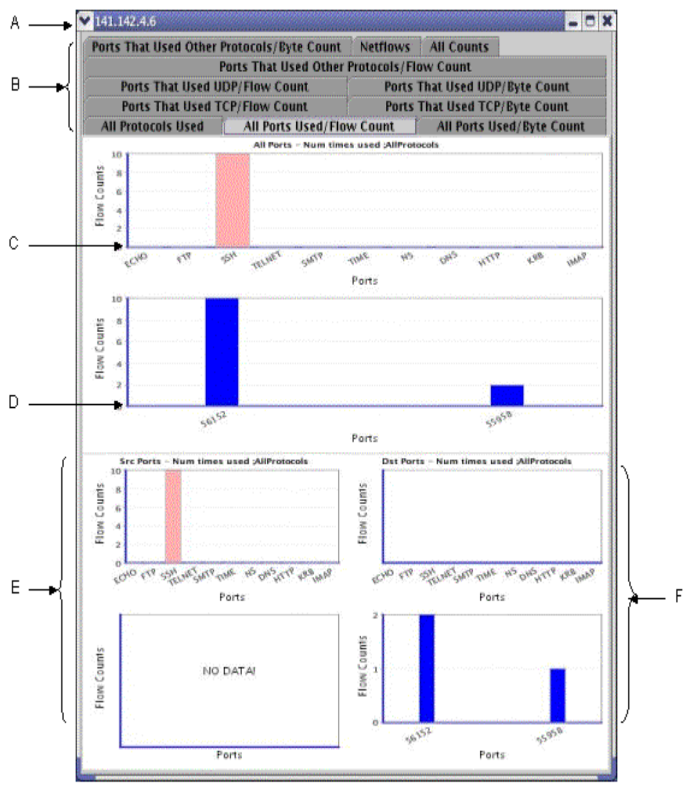
The SMV provides information about adjacent devices in an IP address space. The primary purpose of the SMV is facilitating quick browsing of subnets within an address space for information about the ports and protocols used by each IP device. The user can scan and compare activity across the subset of machines selected using a mouse to highlight the region of interest. Each square in the SMV grid (Figure 4:Label I) represents a device with an IP address. Each square is divided into two histograms: (1) the top histogram represents traffic from well-known ports, and (2) the bottom histogram represents traffic on active ports above port 1024 ordered from most to least active. At a glance, a user sees and compares port activity of different devices. If a machine uses an unusual port, this will be immediately visible. Similar to GV, the user can define the colors associated with the particular ports/protocols. Also, the user can define what ports/protocols are considered "of special interest" using the interface in (Figure 4:Label E).

The MV is the most detailed view simultaneously displaying all statistics from a single machine. Figure 6 shows the eleven tabs that a user may select to view different information from a particular machine. The eleven tabs seen in the MV hold information on the statistics from Figure 2 plus the raw NetFlows source data used to generate the information for the machine being examined (Figure 5). Each tab consists of 6 sets of histograms as shown in Figure 6: lower left is source activity leaving the IP device, lower right is destination activity entering the IP device, and the upper half is an aggregate of traffic activity both entering and leaving each IP device.


141.142.4.6
Ports That Used Other Protocols/Byte Count
Netflows
All Counts
Ports That Used Other Protocols/Flow Count
Ports That Used UDP/Flow Count
Ports That Used UDP/Byte Count
Ports That Used TCP/Flow Count
Ports That Used TCP/Byte Count
All Protocols Used
All Ports Used/Flow Count
All Ports Used/Byte Count

Source IP	Destination IP	Start Time	End Time	Source P...	Destination ...	Bytes	Pro...
141.142.4.6	68.74.79.90	20/Aug...	20/Aug...	22	1804	2240	6
141.142.4.6	68.74.79.90	20/Aug...	20/Aug...	22	1804	13...	6
141.142.4.6	68.74.79.90	20/Aug...	20/Aug...	22	1804	6720	6
141.142.4.6	68.74.79.90	20/Aug...	20/Aug...	22	1804	8960	6
128.174.5.49	141.142.4.6	20/Aug...	20/Aug...	119	55958	33...	6
128.174.5.49	141.142.4.6	20/Aug...	20/Aug...	119	56152	32...	6
128.174.5.49	141.142.4.6	20/Aug...	20/Aug...	119	56152	4382	6
141.142.4.6	146.246.24...	20/Aug...	20/Aug...	22	59706	1344	6
141.142.4.6	12.234.205...	20/Aug...	20/Aug...	22	64773	15...	6
141.142.4.6	12.234.205...	20/Aug...	20/Aug...	22	64773	17...	6
141.142.4.6	12.234.205...	20/Aug...	20/Aug...	22	64773	14...	6
141.142.4.6	12.234.205...	20/Aug...	20/Aug...	22	64850	16...	6
141.142.4.6	12.234.205...	20/Aug...	20/Aug...	22	64926	34...	6

Figure 5. NetFlows Raw Data Tab Within MV


A
B
C
D
E
F
141.142.4.6
Ports That Used Other Protocols/Byte Count
Netflows
All Counts
Ports That Used Other Protocols/Flow Count
Ports That Used UDP/Flow Count
Ports That Used UDP/Byte Count
Ports That Used TCP/Flow Count
Ports That Used TCP/Byte Count
All Protocols Used
All Ports Used/Flow Count
All Ports Used/Byte Count
All Ports - Num times used :AllProtocols
Flow Counts
Ports
Src Ports - Num times used :AllProtocols
Dst Ports - Num times used :AllProtocols
NO DATA!

A. IP address of the machine
B. Different tabs that display information of this machine.
C. All the special ports vs., number of flows it was used in (flow count) this machine
D. All the other ports (All ports *minus* special ports) used by this machine
E. Set of bar charts showing ports that were used as source ports
F. Set of bar charts showing ports that were used as destination ports

Figure 6: NVisionIP Machine View (MV)

IV. Network Management

NVisionIP allows a network administrator to transparently monitor flows to/from each device on a network in order to learn the behavior of the network being managed. This is a different approach than alarming a network with sensors searching for signatures or thresholds for specified events. However, given clues an operator can also interactively configure GV filters to target suspicious activity for further inspection. Examples of activity NVisionIP has been used to detect includes:

- activity on unallocated parts of an address space indicating malicious scans or backscatter from attacks elsewhere
- DoS attacks into/out of a network
- devices infected with worms scanning to propagate showing a large number of connections attempts
- services conforming to official organizational policies
- unusual activity on ports not seen before
- large byte transfers to/from unexpected devices (malware)

The reader is referred to [1,3,5,10] to gain deeper insight into how NVisionIP has been found to help security engineers discover network security attacks.

V. Summary

NVisionIP is designed to help network administrators visually monitor the status of networked devices on IP address spaces. By presenting information visually on one screen with drill-down levels of detail - Galaxy View, Small Multiple View, and Machine View – a user may determine relationships between events at different levels transparently or with the help of filtering. The NVisionIP animation feature within the GV helps users understand how network devices change state over time. The end result is a situational awareness of the current state of networked devices on large and complex IP address spaces as well as a history of how devices came to their current state. The ability to view and interact with device state information on an entire logical IP address spaces is a new capability - to the knowledge of the authors NVisionIP is the only tool that currently provides this capability.

References

1. R. Bearavolu, K. Lakkaraju, W. Yurcik, and H. Raje, "A Visualization Tool for Situational Awareness of Tactical and Strategic Security Events on Large and Complex Computer Networks" *IEEE Military Communications Conference (Milcom)*, 2003.
2. E. Gamma, R. Helm, R. Johnson and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
3. K. Lakkaraju, W. Yurcik, Adam J. Lee, R. Bearavolu, Y. Li and X. Yin, " NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness", *Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC*) in conjunction with *11th ACM Conf. on Computer and Communications Security (CCS)*, 2004.
4. K. Lakkaraju, W. Yurcik, R. Bearavolu, and A.J. Lee, "NVisionIP: An Interactive Network Flow Visualization Tool for Security", *IEEE International Conference on Systems, Man and Cybernetics (SMC*), 2004.
5. K. Lakkaraju, R. Bearavolu and W. Yurcik, "NVisionIP—A Traffic Visualization Tool for Security Analysis of Large and Complex Networks", *Intl. Multiconference on Measurement, Modeling, and Evaluation of Computer-Communication Systems (Performance TOOLS)*, 2003.
6. B. Shneiderman and C. Plaisant, *Designing the User Interface: Strategies for Effective Human-Computer Interaction, 4th edition*, Addison-Wesley, 2005.
7. J. Tidwell, *UI Patterns and Techniques. Retrieved.* <http://time-tripper.com/uipatterns/Overview_Plus_Detail>
8. E. R. Tufte, *The Visual Display of Quantitative Information 2nd edition*, CT:Graphics Press, 2001.
9. W. Yurcik, "The Design of an Imaging Application for Computer Network Security Based on Visual Information Processing," *SPIE Defense and Security Symposium/Visual Information Processing XIII*, 2004.
10. W. Yurcik, K. Lakkaraju, J. Barlow, and J. Rosendale, "A Prototype Tool for Visual Data Mining of Network Traffic for Intrusion Detection", *3rd IEEE International Conference on Data Mining (ICDM) Workshop on Data Mining for Computer Security (DMSEC)*, 2003.
11. W. Yurcik, J. Barlow, K. Lakkaraju, and M. Haberman, "Two Visual Computer Network Security Monitoring Tools Incorporating Operator Interface Requirements," *ACM CHI Workshop on Human-Computer Interaction and Security Systems (HCISEC)*, 2003.

Covert Channel Detection Using Process Query Systems

Vincent Berk, Annarita Giani, George Cybenko
Thayer School of Engineering
Dartmouth College
Hanover,NH
{*vberk, agiani, gvc*}*@dartmouth.edu*

Abstract

In this paper we use traffic analysis to investigate a stealthy form of data exfiltration. We present an approach to detect covert channels based on a Process Query System (PQS), a new type of information retrieval technology in which queries are expressed as process descriptions.

1 Introduction

1.1 Covert Channels

A covert channel between two machines consists of sending and receiving data bypassing the usual intrusion detection techniques. In storage covert channels, data are written to a storage location by one process and then read by another process. The focus of this research is timing covert channels, in which the attacker modulates the time between the packets that are sent. Information is encoded in the inter-packet delays. Suppose an intruder gained access to machine X in our local network and uses this machine to exfiltrate information to machine A. At the receiver side, machine X receives packets with given delays. As the packets flow through the network they traverse a certain number of forwarding devices such as routers, switches, firewalls or repeaters. This equipment has an influence on the delays between packets so that the inter-packet delay at destination might not be exactly the same as at the source. In other words the section of network between sender and receiver acts as a noisy channel. Figure 1 gives a graphical description of the situation.

Suppose an apparatus on our network registers all the inter-packet delays of outgoing communications. Given the chain of delays that are seen, we want to state with a certain probability if a covert channel is being used. The goal is to analyze the inter-packet delays and see if they

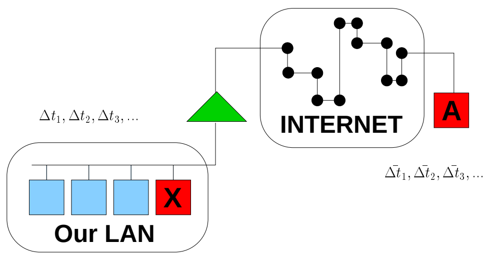

$\Delta t_1, \Delta t_2, \Delta t_3, \ldots$
INTERNET
A
$\bar{\Delta t}_1, \bar{\Delta t}_2, \bar{\Delta t}_3, \ldots$
X
Our LAN

Figure 1: An intruder was able to control machine X which is inside our local network and use it to exfiltrate data coded in inter-packet delays. Machine A is the receiver.

are too particular to be generated by a normal network communication.

The first formal definition of a covert channel was given in [7] as those used for information transmission which are neither designed nor intended to transfer information at all. Later covert channels were defined as those that use entities not normally viewed as data objects but can be manipulated maliciously to transfer information from one subject to another [5, 6]. Since covert communication is very difficult to detect, most researchers resort to investigating methods that simply minimize the amount of information that can be transmitted using a covert timing channel [4, 8].

1.2 Process Query Systems

Process Query Systems are a new paradigm in which user queries are expressed as process descriptions. This allows a PQS to solve large and complex information retrieval problems in dynamic, continually changing environments where sensor input is often unreliable. The system can take input from arbitrary sensors and then forms hypotheses regarding the observed environment, based on the process queries given by the user. Figure 2 shows a simple example of such a model. Model $\mathcal{M}_1$

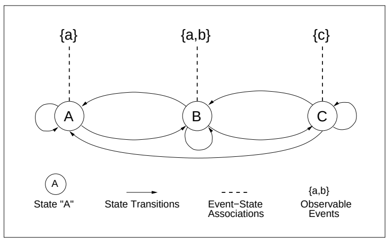


Figure 2: A Simple Process Model, $\mathcal{M}_1$

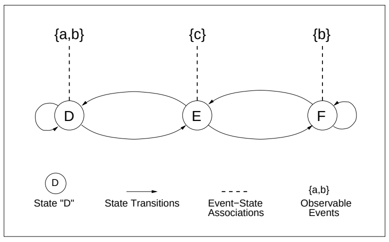


Figure 3: Another Process Model, $\mathcal{M}_2$

represents a state machine $S_1 = (Q_1, \Sigma_1, \delta_1)$, where the set of states $Q_1 = \{A, B, C\}$, the set of observable events $\Sigma_1 = \{a, b, c\}$, and the set of possible associations $\delta_1 : Q_1 \times \Sigma_1$ consists of

$$\delta_1 = \{\{A, a\}, \{B, a\}, \{B, b\}, \{C, c\}\}$$

A possible event sequence recognized by this model would be: $e_1 = a, e_2 = a, e_3 = b, e_4 = c, e_5 = b$ which we will write as $e_{1:5} = aabcb$ for convenience. Possible state sequences that match this sequence of observed events could be $AABCB$, or $ABBCB$, both of which are equally likely given $\mathcal{M}_1$. A rule-based model would need a lot of rules to identify this process, based on all the possible event sequences. Below is a set of all the rules necessary for detecting single transitions:

$$\begin{aligned}
AA \rightarrow \quad & \{aa\} \\
AB \rightarrow \quad & \{aa\}, \{ab\} \\
BB \rightarrow \quad & \{aa\}, \{ab\}, \{ba\}, \{bb\} \\
BA \rightarrow \quad & \{aa\}, \{ba\} \\
BC \rightarrow \quad & \{ac\}, \{bc\} \\
CC \rightarrow \quad & \{cc\} \\
CB \rightarrow \quad & \{ca\}, \{cb\} \\
CA \rightarrow \quad & \{ca\}
\end{aligned}$$

Let us introduce a second model $\mathcal{M}_2$ in Figure 3. Now consider the following sequence of events:

$$e_{1:24} = abaacabbacabacccabacabbc$$

where each observation may have been produced by instances of model $\mathcal{M}_1$, model $\mathcal{M}_2$, or be totally unrelated. A Process Query System uses multiple hypothesis, multiple model techniques to disambiguate observed events and associate them with a "best fit" description of which processes are occurring and in what state they are. In comparison, a rule-based system would get impossibly complex for the above situation.

A PQS is a very general and flexible core that can be applied to many different fields. The only things that change between different applications of a PQS are the format of the incoming observation stream(s) and the submitted model(s). Internally a PQS has four major components that are linked in the following order:

1. Incoming observations.
2. Multiple hypothesis generation.
3. Hypothesis evaluation by the models.
4. Selection, pruning, and publication.

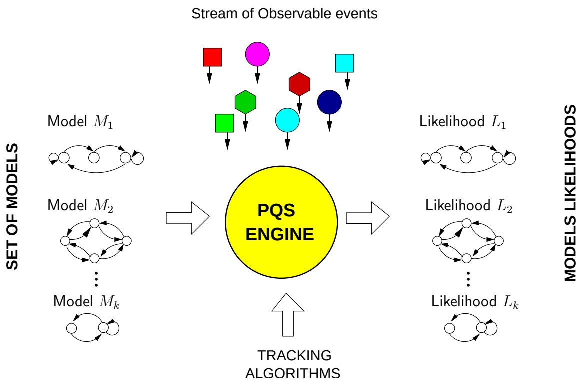


Figure 4: A set of models is given as input to the engine. Given a stream of observations and using tracking algorithms the engine returns the likelihood of each model.

The big benefits of a PQS are its superior scalability and applicability. The application programmer simply connects the input event streams and then focuses on writing process models. Models can be constructed as state machines, formal language descriptions, Hidden Markov Models, kinematic descriptions, or a set of rules. The PQS is now ready to track processes occurring in a dynamic environment and continuously present *the best possible explanation of the observed events* to the user.

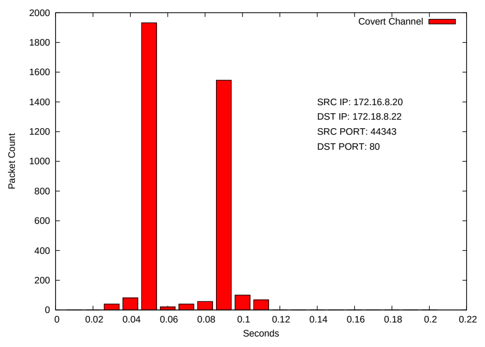
See Figure 4 for a graphical representation of the system. A detailed overview and application of a Process Query System can be found in [2, 1, 3].

2 Detection

2.1 Covert Channel Models

We focus our attention on binary codes. Our models are based on the assumption that in the case of a covert channel, the inter-packet delays will center around two distinct values (ie. two distinct delays), see Figure 5 (a).

(a)



(b)

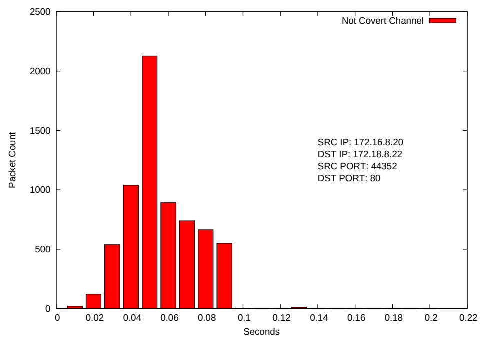


Figure 5: The figures show the number of packets received with a given delay. Horizontal axis shows the inter-arrival time in seconds, vertical axis shows the number of packet arrived. The parameters are the following. In case (a) the two spikes show that a covert channel communication is in place. Case (b) represents normal communication.

For a covert channel the sample mean μ (average) of the inter-packet delays will be somewhere between the two spikes. The packet-count in the histogram at that point will therefore be very low. However, looking at a normal traffic pattern the mean of the inter-packet delays will be in the center of the large spike. The packet-count at the mean will thus be very high, if not the highest. If we divide the packet-count at the mean by the maximum packet-count from the histogram, we get a measure of how likely it is that the communication is a covert channel. In particular the smaller is the ratio $\frac{C_\mu}{C_{max}}$ the higher is the probability of having a covert channel communication. We can therefore assume the following probability:

$$P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$$

where $C(\mu)$ is the packet-count at the mean and C_{max} is the maximum packet-count of the histogram. Experiments with three different types of data were conducted, and Figure 6 shows the ratio $\frac{C_\mu}{C_{max}}$ for these experiments. The data considered are of the following types.

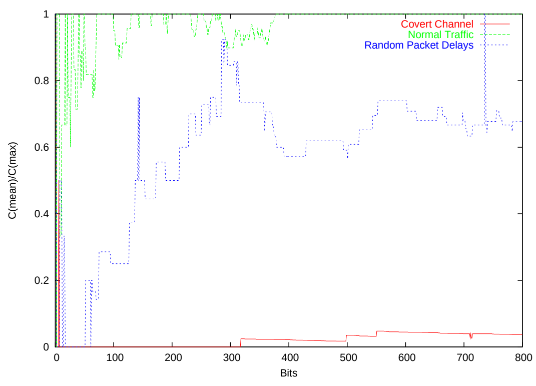


Figure 6: The ratio between the packet count at the sample mean and the maximum packet count for normal traffic, fully random delays, and for a binary covert channel. Horizontal axis shows the length of the estimated sequence in bits, vertical axis shows $\frac{C_\mu}{C_{max}}$.

Normal Data. Packets with a average delay of 0.2 seconds were transmitted. The inter-arrival times vary, but the maximum number of packets has a delay of 0.2 seconds. The sample mean μ is therefore represented by a delay very close to 0.2 and the number of packets with exactly that delay (C_μ) is very high. The ratio between this number and the maximum number of a packet for a certain delay (C_{max}) quickly grows to 1.0, and stays there as more packets are transmitted.

Random Data. Packets are sent with a fully random delay. Although this is not realistic for traffic encountered on the network, it does present a good idea of the worst case scenario. Initially, when only a few bits have been sent, the delays scatter across the range and it is unlikely that the sample mean will have a high count. That explains why until approximately the first 10 bytes the ratio $\frac{C_\mu}{C_{max}}$ remains zero. Later on, as more packets arrive the histogram starts to flatten so the ratio starts to rise. As the number of transmitted packets increases even further the ratio keeps growing, until it eventually hits 1.0 as the packet count goes to infinity.

Covert Channel Communication. Two delays are used, thus the interarrival times concentrate around those two values. The sample mean μ lies approximately in the middle between the two spikes. The count C_μ is low and therefore the ratio $\frac{C_\mu}{C_{max}}$ is approximately zero. As more and more bits are transmitted over the covert channel the spikes increase in size, however that ratio always remains very close to zero.

Our algorithm detects the sequence that most likely represents the covert communication channel analyzing the value $\frac{C_\mu}{C_{max}}$. The lower that value the higher is the probability of having a malicious communication hidden in inter-packet delays.

Figure 6 compares the three cases.

2.2 Implementation of a PQS

Our current implementation of a Process Query System is called TRAFEN (TRacking And Fusion ENgine). We implemented software applications to use as sensors that send data to the engine. TRAFEN then, given such observations, returns the most likely hypotheses.

In order to input observations in the system we built sensors. This software monitors the packet flows and returns quantities that are crucial to our models. It aggregates packets that have the same source IP, destination IP, source Port, destination Port and computes the quantity $\frac{C_\mu}{C_{max}}$. An observation looks like the following:

Src ip, Src port,
Dst ip, Dst port,
Protocol, $\frac{C_\mu}{C_{max}}$.

The TRAFEN engine evaluates each incoming observation and checks their correlation. Correlated events are assigned to the same track. Sets of tracks constitute a hypothesis. At the moment our process descriptions are mainly based on first principles. An example of a model used in our experiments is the following.

Singleton: Get observation

$$\frac{C_\mu}{c_{max}} \geq 0.9 \Rightarrow Score = 0.001$$

$$\frac{C_\mu}{c_{max}} < 0.9 \Rightarrow Score = 0$$

Track: Get observation OBS. If OBS is consistent (same characteristics with the track)

$$f = 1 - \frac{1}{lenght\ track + 1}$$

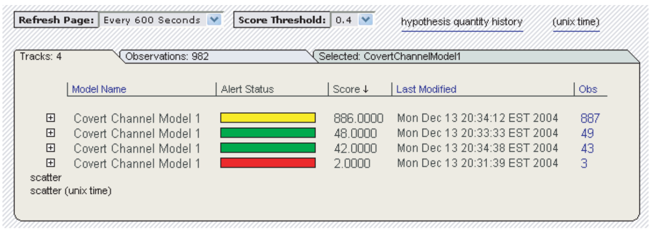
$$Track\ Score = f \cdot \left(1 - \frac{C_{\mu\ Track}}{C_{max\ Track}}\right)$$

If OBS is NOT consistent with the track $Track\ Score = 0$.

The first case considers only tracks with one observation. The score is dependent only on the ratio. In the second case tracks with more than one observation are considered. In this case we assume that the longer the track is, the higher the score will be.

3 Results

A covert channel communication was simulated and many models were built and inserted into TRAFEN. We found that some models performed better than others. All of them were able to detect covert channel communication but some of them returned also false positive. We built a front-end displaying the most likely hypothesis, the different tracks together with their score, see Figure 7.


Refresh Page: Every 600 Seconds
Score Threshold: 0.4
hypothesis quantity history
(unix time)
Tracks: 4
Observations: 982
Selected: CovertChannelModel1

	Model Name	Alert Status	Score ↓	Last Modified	Obs
⊞	Covert Channel Model 1	🟨	886.0000	Mon Dec 13 20:34:12 EST 2004	887
⊞	Covert Channel Model 1	🟩	48.0000	Mon Dec 13 20:33:33 EST 2004	49
⊞	Covert Channel Model 1	🟩	42.0000	Mon Dec 13 20:34:38 EST 2004	43
⊞	Covert Channel Model 1	🟥	2.0000	Mon Dec 13 20:31:39 EST 2004	3

scatter
scatter (unix time)

Figure 7: www.pqsnet.net/display.

4 Conclusion and Future Work

Process query system is a crucial tool in the analysis and detection of network threats. We applied it to solve the problem of detecting covert channel and we found very promising results. We plan to investigate different types of exfiltration of information and develop detection techniques.

5 Acknowledgments

Supported under ARDA P2INGS Award No. F30602-03-C-0248. Points of view in this document are those of the author(s) and do not necessarily represent the official position of ARDA.

References

[1] V. Berk and N. Fox. Process Query Systems for Network Security Monitoring. *Proceedings of the SPIE*, March, 2005.

[2] G. Cybenko, V. Berk, V. Crespi, R. Gray, and G. Jiang. An Overview of Process Query systems. *Proceedings of the SPIE*, April, 2003.

[3] G. Cybenko, V. Berk, and C. Roblee. Large-Scale Autonomic Server Monitoring Using Process Query Systems. *Proceedings of the SPIE*, March, 2005.

[4] J. Giles and B. Hajek. An Information-Theoretic and Game-theoretic Study of Timing Channels. *IEEE Trans. on Information Theory*, 48(9):2455–2477, Sept 2002.

[5] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel. *ACM Transaction on Computer Systems*, 1(3):256–277, Aug 1983.

[6] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel : Twenty years later. *Proc. 18th Annual Computer Security Applications Conference (ACSAC)*, pages 109–118, 2002.

[7] B. W. Lampson. A Note on the Confinment Problem. *Proc. of the Communication of the ACM*, 16(10):613–615, Oct 1973.

[8] N. Ogurtsov, H. Orman, R. Schroeppel, S. O'Malley, and O. Spatscheck. Covert Channel Elimination Protocols. *Technical Reports TR96-14. Department of Computer Science, University of Arizona*, 1996.

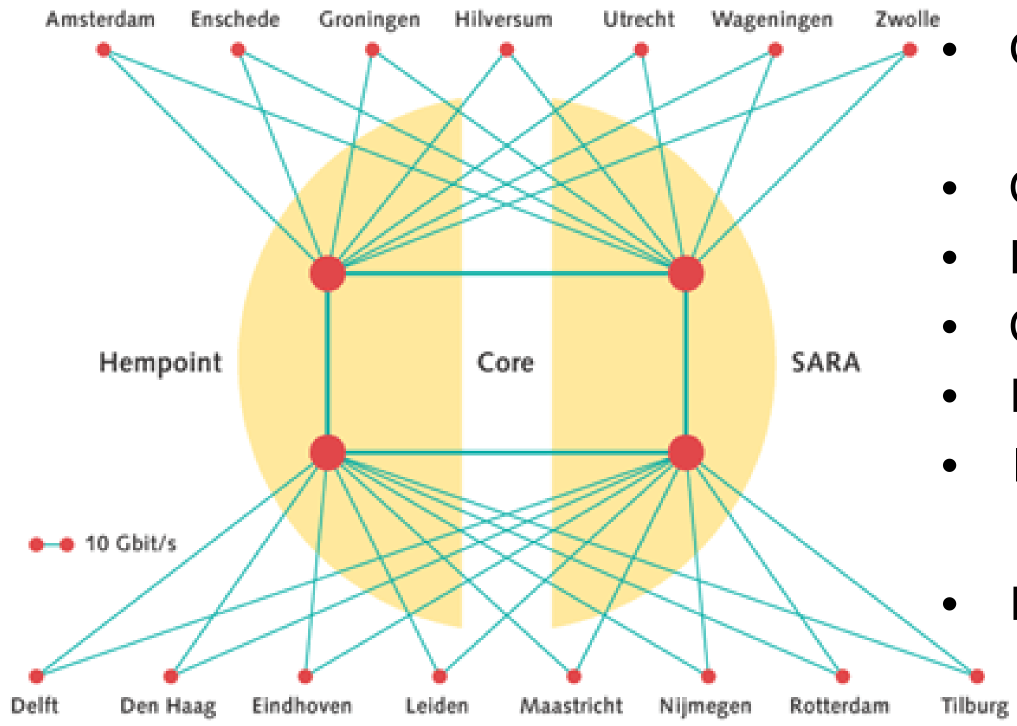
SURFnet

NERD: Network Emergency Responder & Detector

Wim.Biemolt@surfnet.nl

2nd FloCon, Pittsburgh, September, 2005.

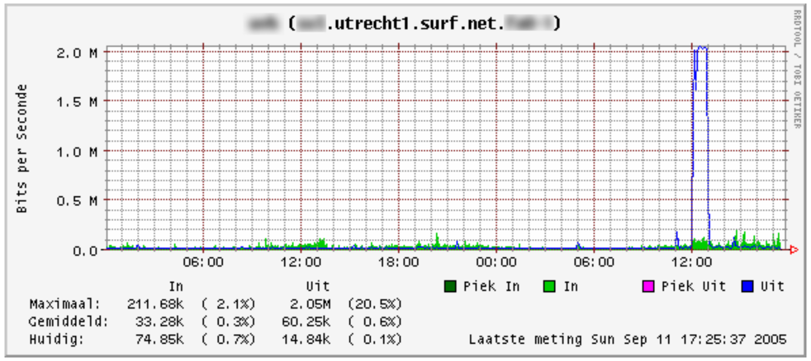
SURFnet5 network



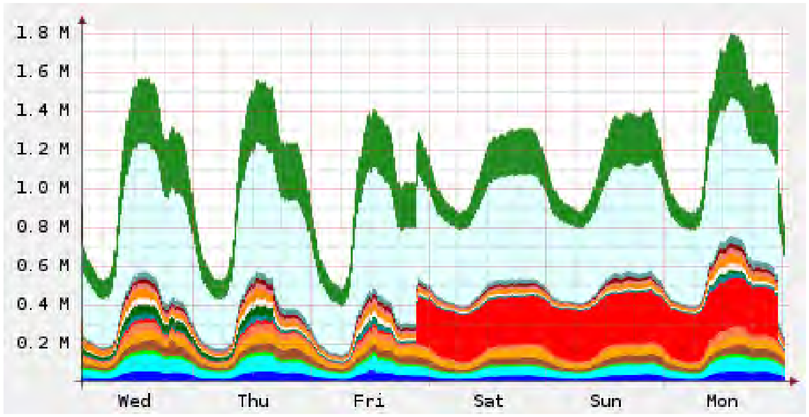
- Operational
  - Since September 2001
- Cisco 12416 routers
- Backbone: 10Gbps
- Connections: 1Gbps
- Dual stack (6PE)
- Incident detection
  - SURFnet & TNO: 2002
- Decommissioning
  - End of December 2005

Incident response tools

- SURFstat
  - mrtg/rrdtool
- Research
  - syslog
  - Netflow
    - promising at the required speeds (>10 Gbps)
    - sampled (`ip flow-sampling-mode packet-interval 100`)
  - Full data analysis requires high-end equipment
- Prototype
  - cflowd (caida)
    - no longer supported
  - gnuplot, mysql, php
  - Not open-source

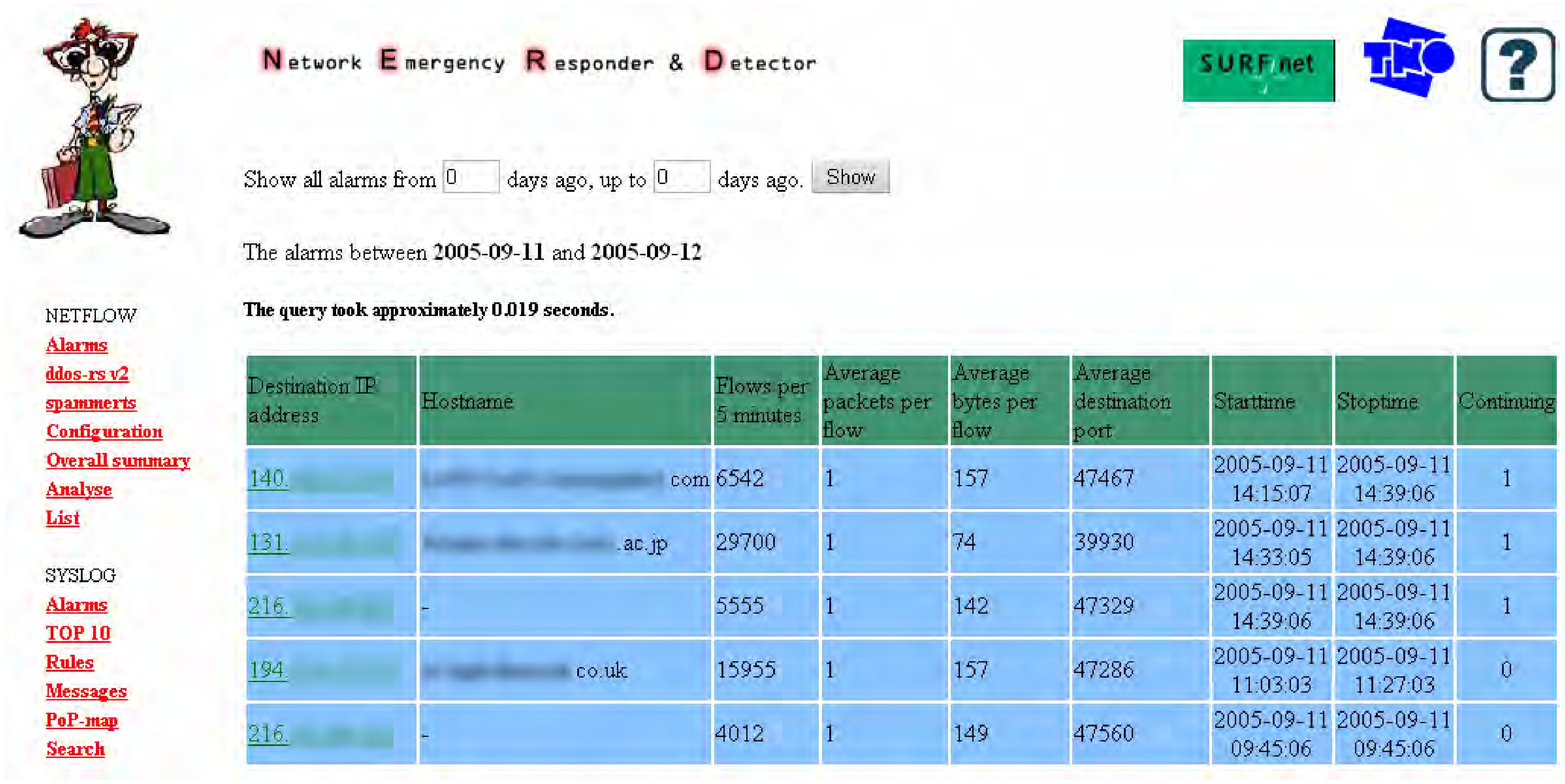

[redacted] ([redacted].utrecht1.surf.net.[redacted])

```
                In                 Uit
Maximaal:    211.68k  ( 2.1%)    2.05M  (20.5%)
Gemiddeld:    33.28k  ( 0.3%)   60.25k  ( 0.6%)
Huidig:       74.85k  ( 0.7%)   14.84k  ( 0.1%)       Laatste meting Sun Sep 11 17:25:37 2005
```



SURFnet

Prototype



NETFLOW
Alarms
ddos-rs v2
spammerts
Configuration
Overall summary
Analyse
List

SYSLOG
Alarms
TOP 10
Rules
Messages
PoP-map
Search

Show all alarms from 0 days ago, up to 0 days ago. Show

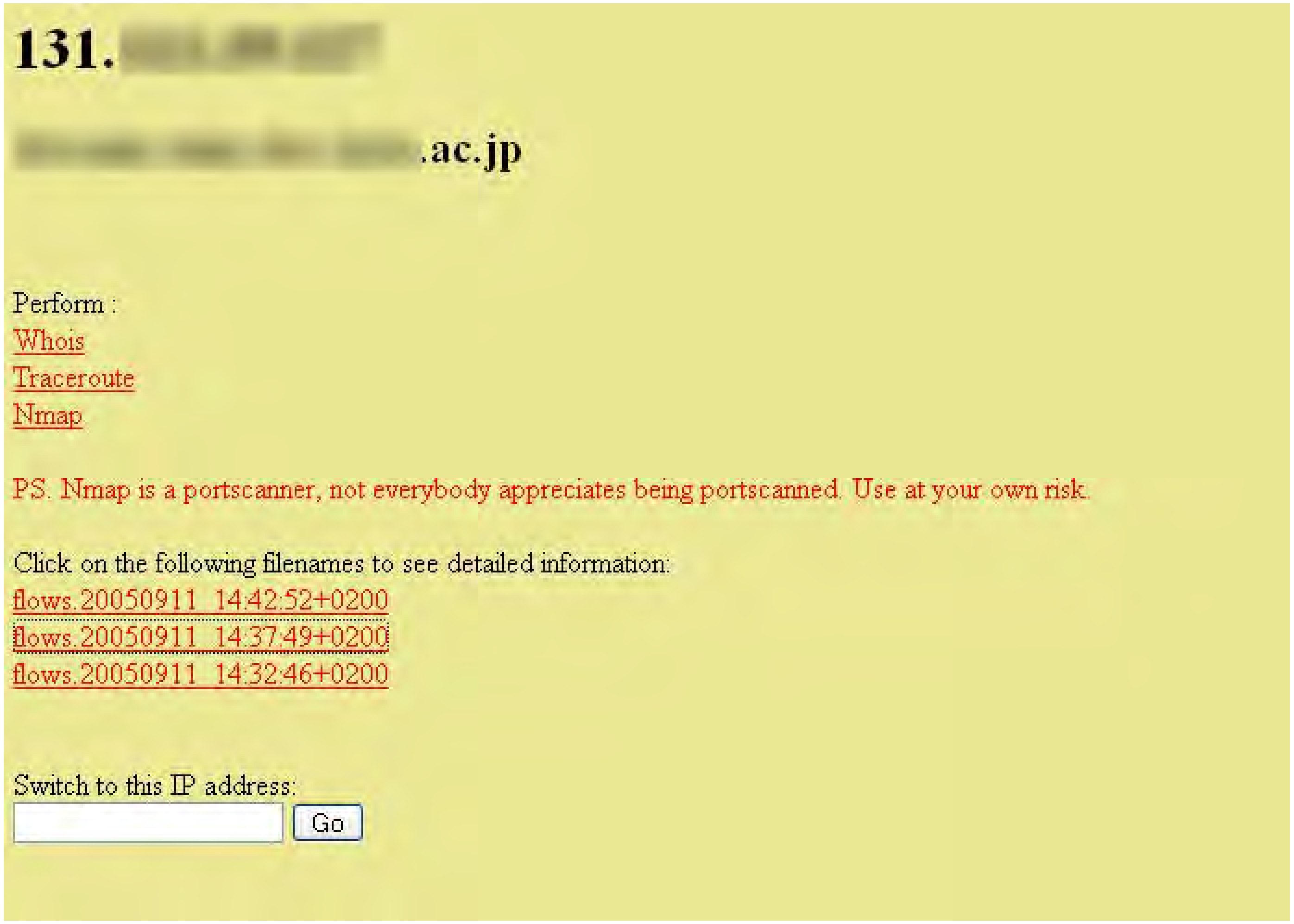
The alarms between **2005-09-11** and **2005-09-12**

The query took approximately 0.019 seconds.

Destination IP address	Hostname	Flows per 5 minutes	Average packets per flow	Average bytes per flow	Average destination port	Starttime	Stoptime	Continuing
140. [redacted]	[redacted] com	6542	1	157	47467	2005-09-11 14:15:07	2005-09-11 14:39:06	1
131. [redacted]	[redacted] .ac.jp	29700	1	74	39930	2005-09-11 14:33:05	2005-09-11 14:39:06	1
216. [redacted]	-	5555	1	142	47329	2005-09-11 14:39:06	2005-09-11 14:39:06	1
194. [redacted]	[redacted] co.uk	15955	1	157	47286	2005-09-11 11:03:03	2005-09-11 11:27:03	0
216. [redacted]	-	4012	1	149	47560	2005-09-11 09:45:06	2005-09-11 09:45:06	0

SURFnet

Alarm



131.[redacted]

[redacted].ac.jp

Perform :
Whois
Traceroute
Nmap

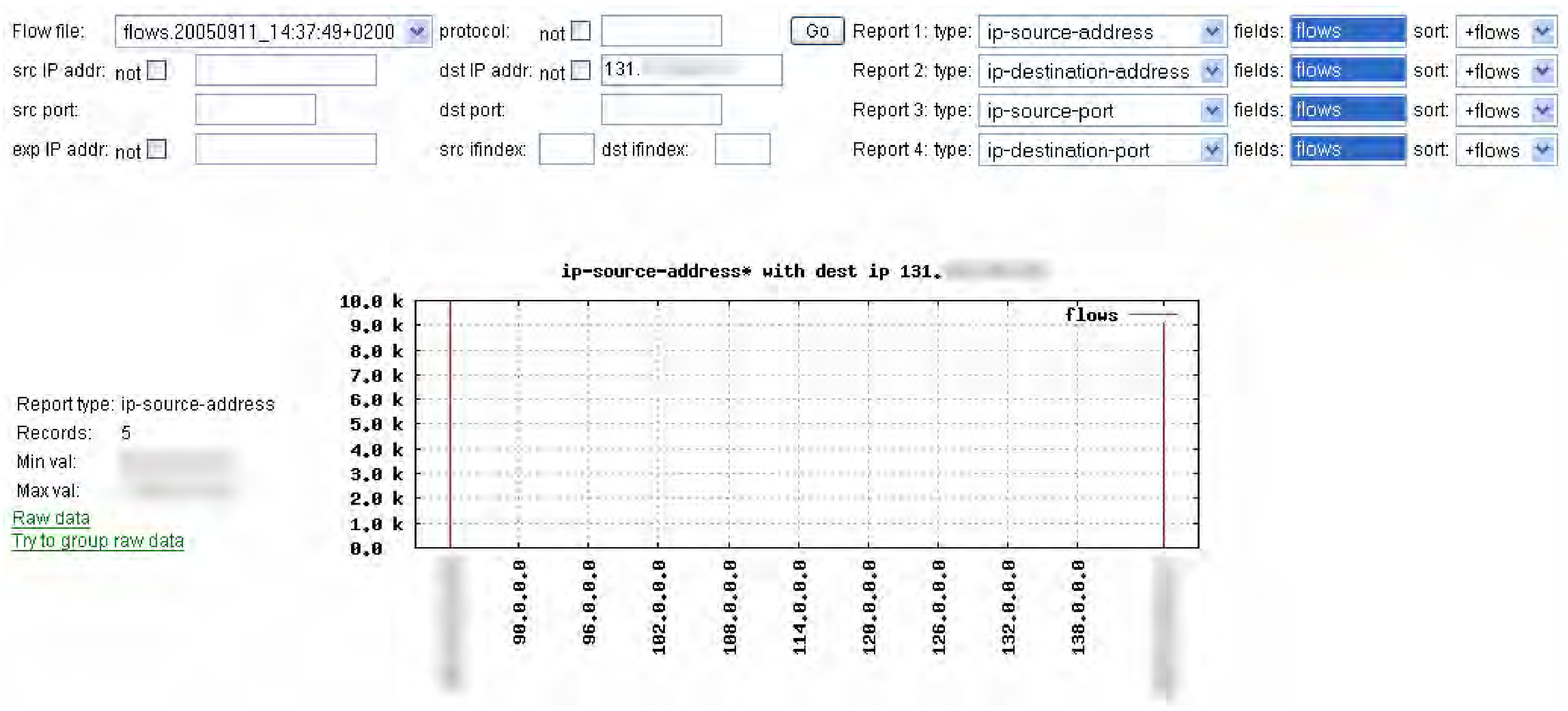
PS. Nmap is a portscanner, not everybody appreciates being portscanned. Use at your own risk.

Click on the following filenames to see detailed information:
flows.20050911_14:42:52+0200
flows.20050911_14:37:49+0200
flows.20050911_14:32:46+0200

Switch to this IP address:

SURFnet

Analyse



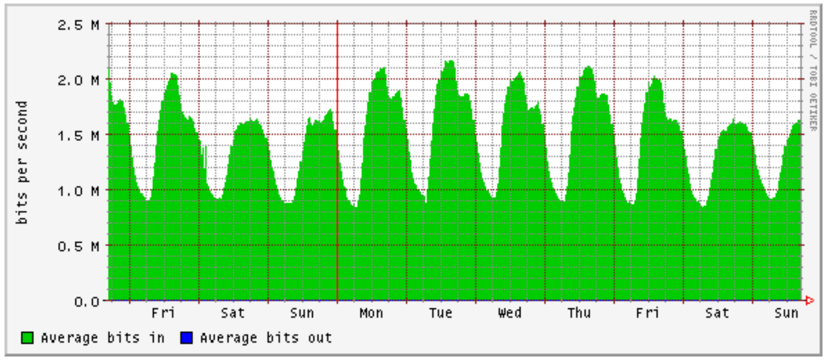
Flow file:	flows.20050911_14:37:49+0200	protocol: not ☐		Go	Report 1: type:	ip-source-address	fields: flows	sort: +flows
src IP addr: not ☐		dst IP addr: not ☐	131.[redacted]		Report 2: type:	ip-destination-address	fields: flows	sort: +flows
src port:		dst port:			Report 3: type:	ip-source-port	fields: flows	sort: +flows
exp IP addr: not ☐		src ifindex:	dst ifindex:		Report 4: type:	ip-destination-port	fields: flows	sort: +flows

Report type: ip-source-address
Records: 5
Min val: [redacted]
Max val: [redacted]
Raw data
Try to group raw data

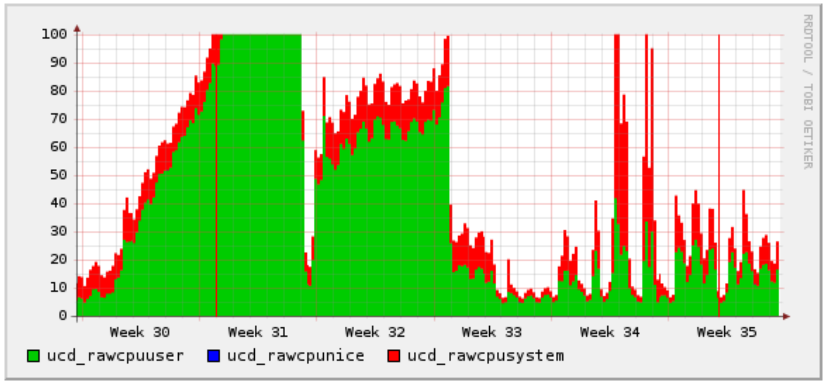
SURFnet

Hardware

- Dell PowerEdge 1650
  - 04-2002, RedHat 7
  - 1x 1.4GHz, 1GB, 3x 36GB
- Dell PowerEdge 2650
  - 12-2003, FreeBSD 4.11
  - 2x 3GHz, 4GB, 5x 146GB
- Dell PowerEdge 2850
  - 10-2004, FreeBSD 5.4
  - 2x 3.4GHz, 6GB, 6x 146GB
- Dell PowerEdge 2850
  - 06-2005, FreeBSD 6.0
  - 2x 3.6GHz, 4GB, 6x 300GB
- SunFire V240
  - 12-2004, Solaris 10
  - 2x 1.5GHz, 4GB, 4x 146GB



http://www.switch.ch/tf-tant/floma/sw/samplicator/



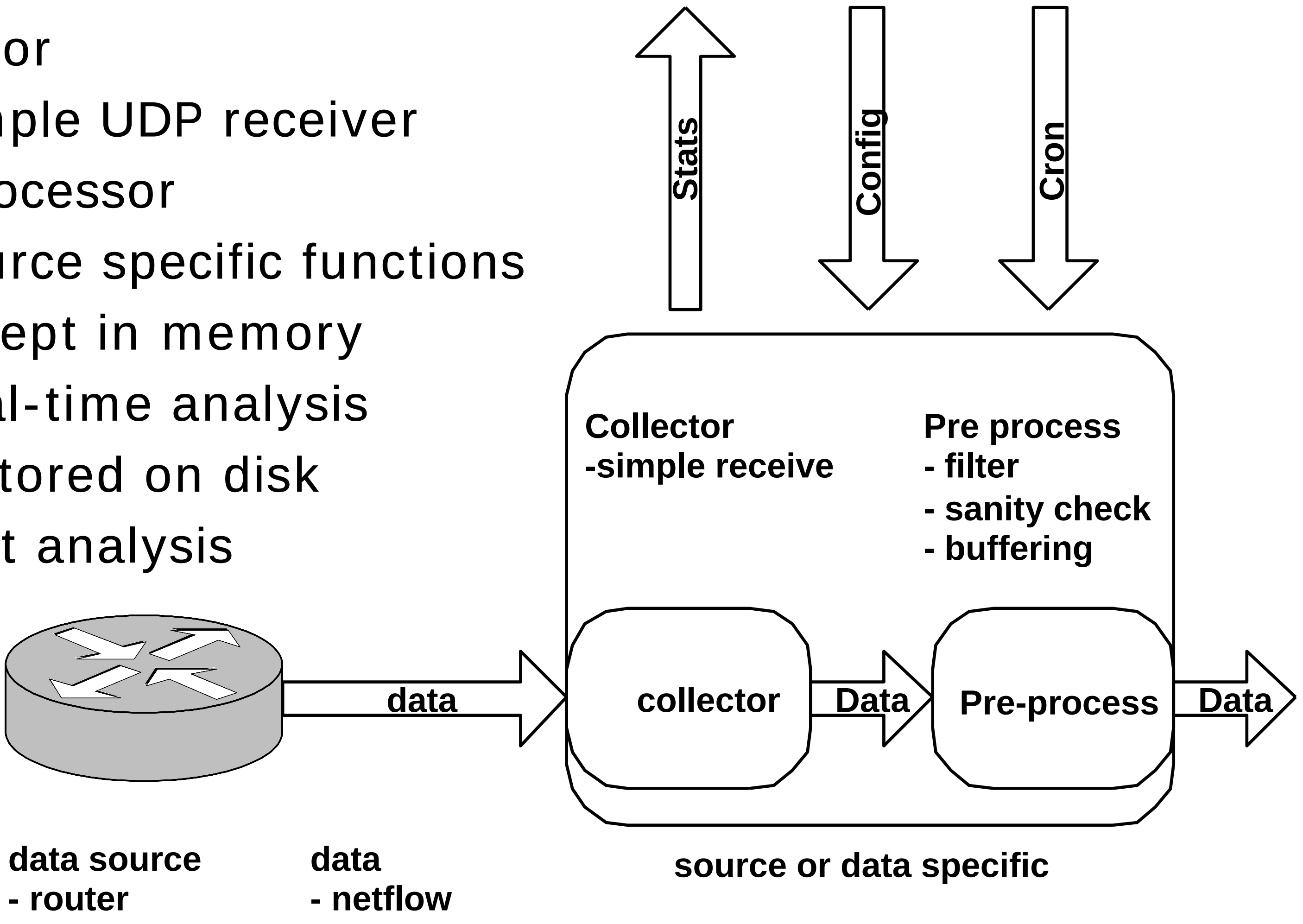
Some specs of the new NERD

- nerdd, analysis
  - boost libraries, MySQL database, php, plplot
- Netflow versions
  - V5 (tested)
  - V9 (IPFIX)
- Platforms tested
  - FreeBSD
  - Linux
- Apache Open Source Licence v2.0

SURFnet

Software Architecture

- Collector
  - Simple UDP receiver
- Pre-processor
  - Source specific functions
- Data kept in memory
  - Real-time analysis
- Data stored on disk
  - Post analysis



Real-time and post analysis

- Real time analysis
  - Rules can be used for ‘real-time’ analysis
    - A rule is a combination of filters, clusters and a threshold for some metric (e.g. number of flows)
  - Example of a rule
    - Filter “port=445”, cluster “dst IP”, threshold=1000 flows/min
  - Results in an alarm if a host receives more then 1000 flows/min on TCP port 445
  - Output formatting: alarm in database
  - Every x minutes the rules (1…n) are executed
- Post analysis
  - Executed at user request
  - Rules without threshold
  - Output formatting: flow-tools like text file, graphical output

Functionality – Filters & Clusters

- Sample of Netflow data

src	prt	dst	prt
10.0.0.1	2000	10.0.0.2	23
10.0.0.3	1000	10.0.0.2	22
10.0.0.6	2000	10.0.0.2	22
10.0.0.1	1000	10.0.0.3	23
10.0.0.1	1000	10.0.0.3	23

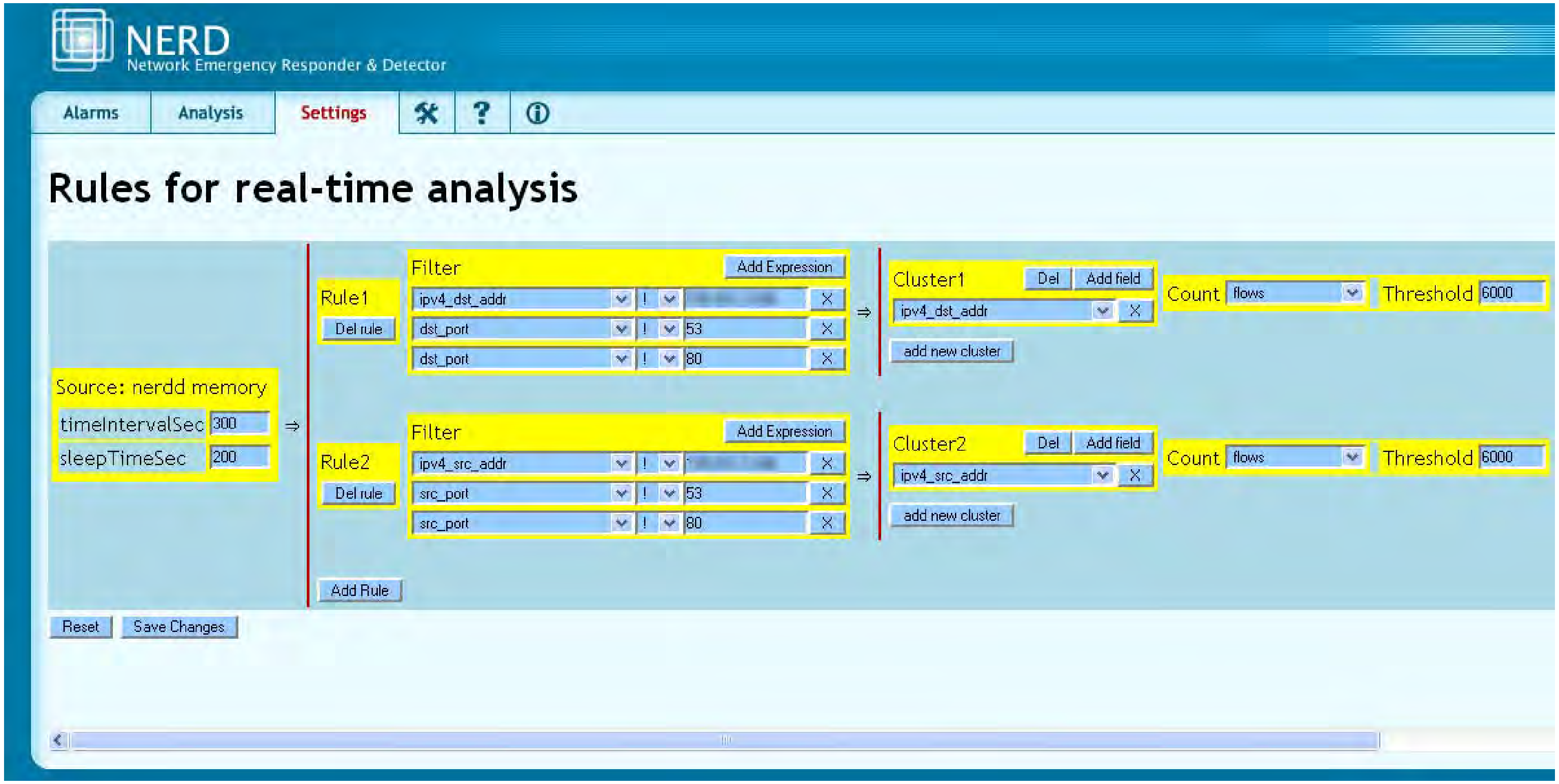
- Example: filter “src port=2000”

src	prt	dst	prt
10.0.0.1	2000	10.0.0.2	23
10.0.0.6	2000	10.0.0.2	22

- Example: filter, cluster “dst port” & count flows

prt	# of flows
22	1
23	1

Real-time analysis - configuration



Rules for real-time analysis

Source: nerdd memory
timeIntervalSec 300
sleepTimeSec 200

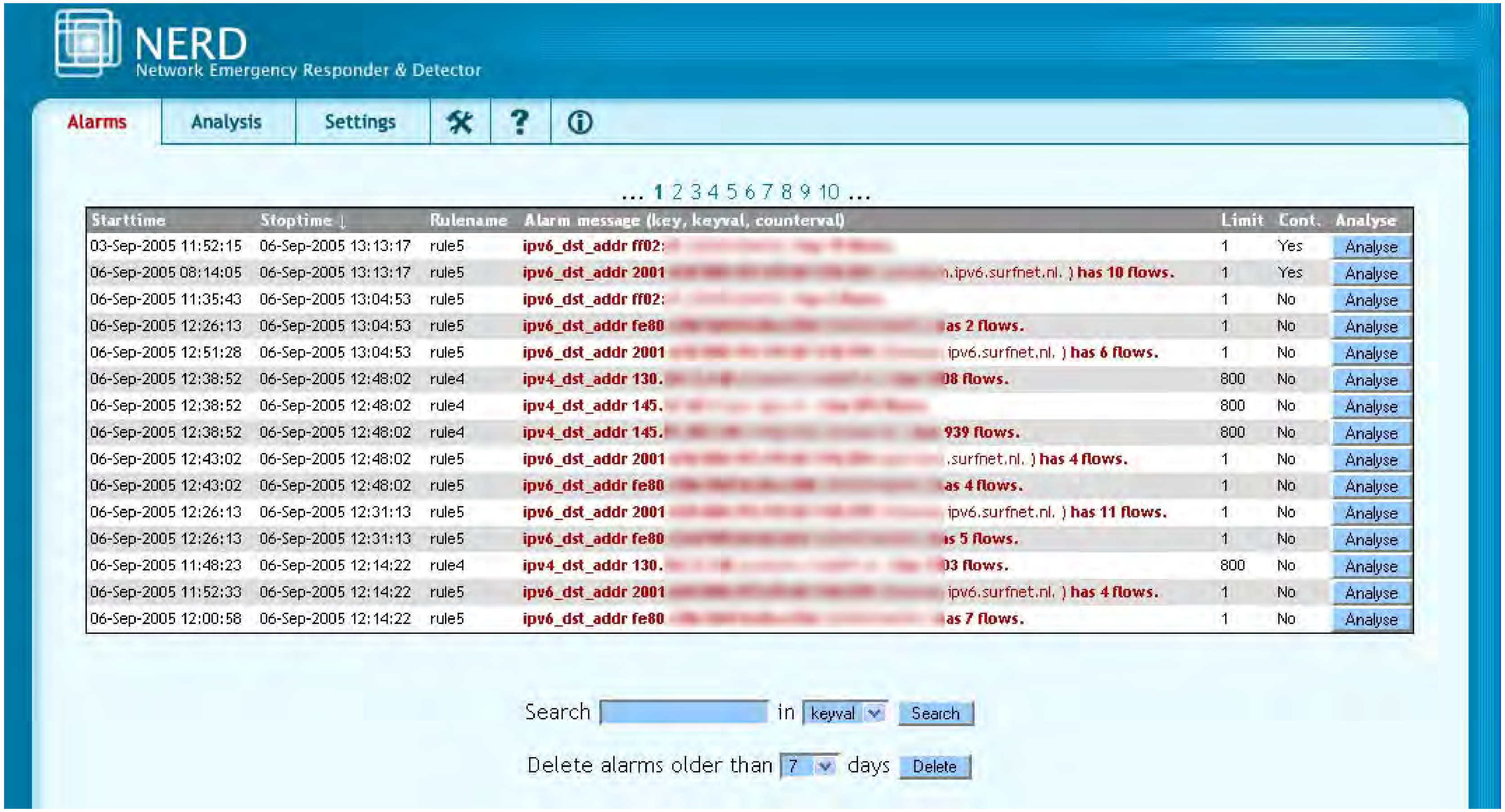
Rule1 (Del rule)
Filter (Add Expression)
ipv4_dst_addr ! [redacted]
dst_port ! 53
dst_port ! 80
⇒ Cluster1 (Del, Add field): ipv4_dst_addr (add new cluster)
Count flows
Threshold 6000

Rule2 (Del rule)
Filter (Add Expression)
ipv4_src_addr ! [redacted]
src_port ! 53
src_port ! 80
⇒ Cluster2 (Del, Add field): ipv4_src_addr (add new cluster)
Count flows
Threshold 6000

Add Rule

Reset | Save Changes

Alarms

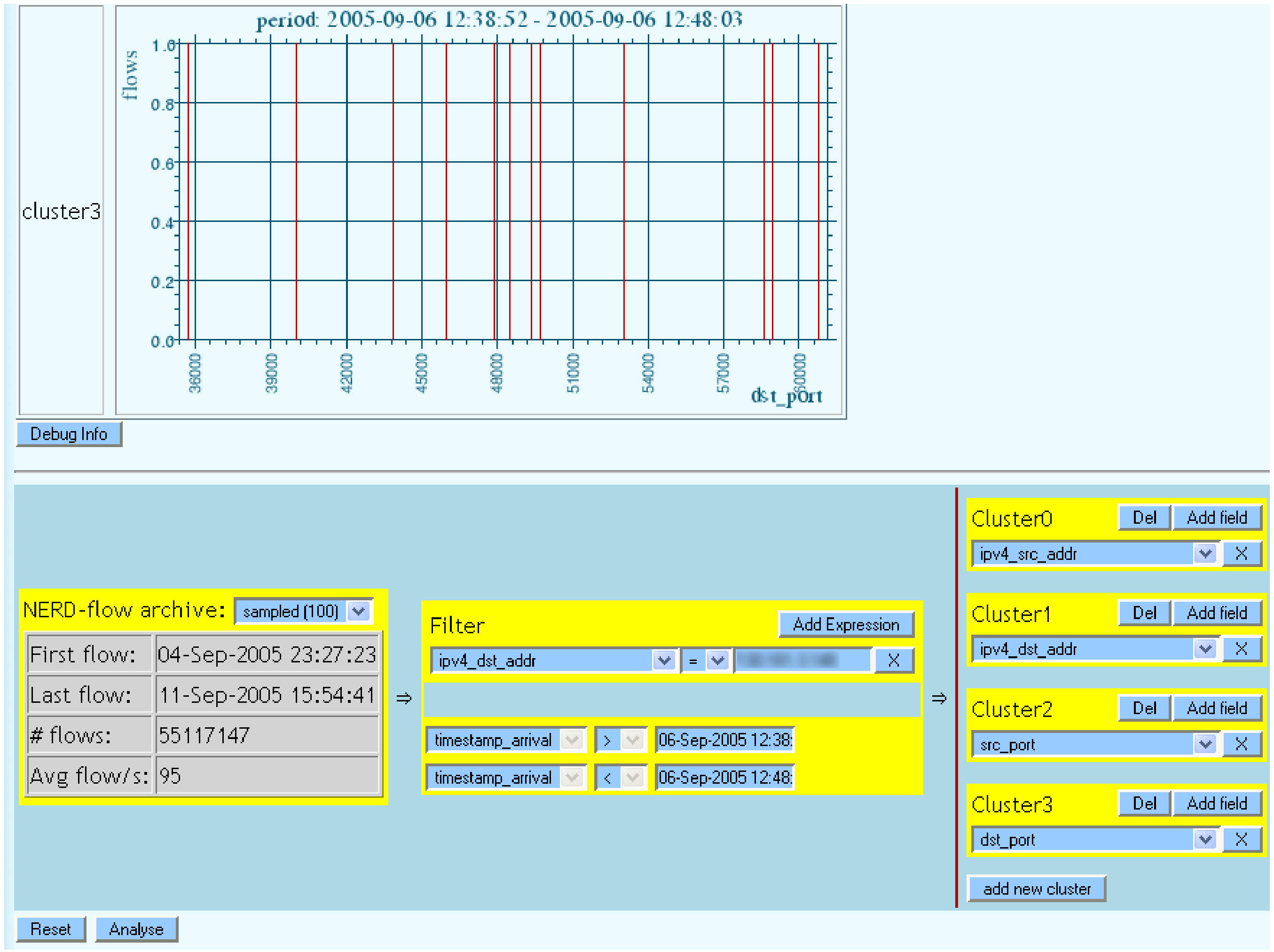


Starttime	Stoptime ↓	Rulename	Alarm message (key, keyval, counterval)	Limit	Cont.	Analyse
03-Sep-2005 11:52:15	06-Sep-2005 13:13:17	rule5	ipv6_dst_addr ff02: [redacted]	1	Yes	Analyse
06-Sep-2005 08:14:05	06-Sep-2005 13:13:17	rule5	ipv6_dst_addr 2001 [redacted].ipv6.surfnet.nl.) has 10 flows.	1	Yes	Analyse
06-Sep-2005 11:35:43	06-Sep-2005 13:04:53	rule5	ipv6_dst_addr ff02: [redacted]	1	No	Analyse
06-Sep-2005 12:26:13	06-Sep-2005 13:04:53	rule5	ipv6_dst_addr fe80 [redacted]as 2 flows.	1	No	Analyse
06-Sep-2005 12:51:28	06-Sep-2005 13:04:53	rule5	ipv6_dst_addr 2001 [redacted] ipv6.surfnet.nl.) has 6 flows.	1	No	Analyse
06-Sep-2005 12:38:52	06-Sep-2005 12:48:02	rule4	ipv4_dst_addr 130. [redacted]08 flows.	800	No	Analyse
06-Sep-2005 12:38:52	06-Sep-2005 12:48:02	rule4	ipv4_dst_addr 145. [redacted]	800	No	Analyse
06-Sep-2005 12:38:52	06-Sep-2005 12:48:02	rule4	ipv4_dst_addr 145. [redacted] 939 flows.	800	No	Analyse
06-Sep-2005 12:43:02	06-Sep-2005 12:48:02	rule5	ipv6_dst_addr 2001 [redacted] .surfnet.nl.) has 4 flows.	1	No	Analyse
06-Sep-2005 12:43:02	06-Sep-2005 12:48:02	rule5	ipv6_dst_addr fe80 [redacted]as 4 flows.	1	No	Analyse
06-Sep-2005 12:26:13	06-Sep-2005 12:31:13	rule5	ipv6_dst_addr 2001 [redacted] ipv6.surfnet.nl.) has 11 flows.	1	No	Analyse
06-Sep-2005 12:26:13	06-Sep-2005 12:31:13	rule5	ipv6_dst_addr fe80 [redacted]s 5 flows.	1	No	Analyse
06-Sep-2005 11:48:23	06-Sep-2005 12:14:22	rule4	ipv4_dst_addr 130. [redacted]03 flows.	800	No	Analyse
06-Sep-2005 11:52:33	06-Sep-2005 12:14:22	rule5	ipv6_dst_addr 2001 [redacted] ipv6.surfnet.nl.) has 4 flows.	1	No	Analyse
06-Sep-2005 12:00:58	06-Sep-2005 12:14:22	rule5	ipv6_dst_addr fe80 [redacted]as 7 flows.	1	No	Analyse

Search [ ] in keyval Search

Delete alarms older than 7 days Delete

Analysis – IPv4



NERD-flow archive: sampled (100)

First flow:	04-Sep-2005 23:27:23
Last flow:	11-Sep-2005 15:54:41
# flows:	55117147
Avg flow/s:	95

⇒

Filter — Add Expression

ipv4_dst_addr = [redacted] X

timestamp_arrival > 06-Sep-2005 12:38:

timestamp_arrival < 06-Sep-2005 12:48:

⇒

Cluster0 — Del — Add field
ipv4_src_addr X

Cluster1 — Del — Add field
ipv4_dst_addr X

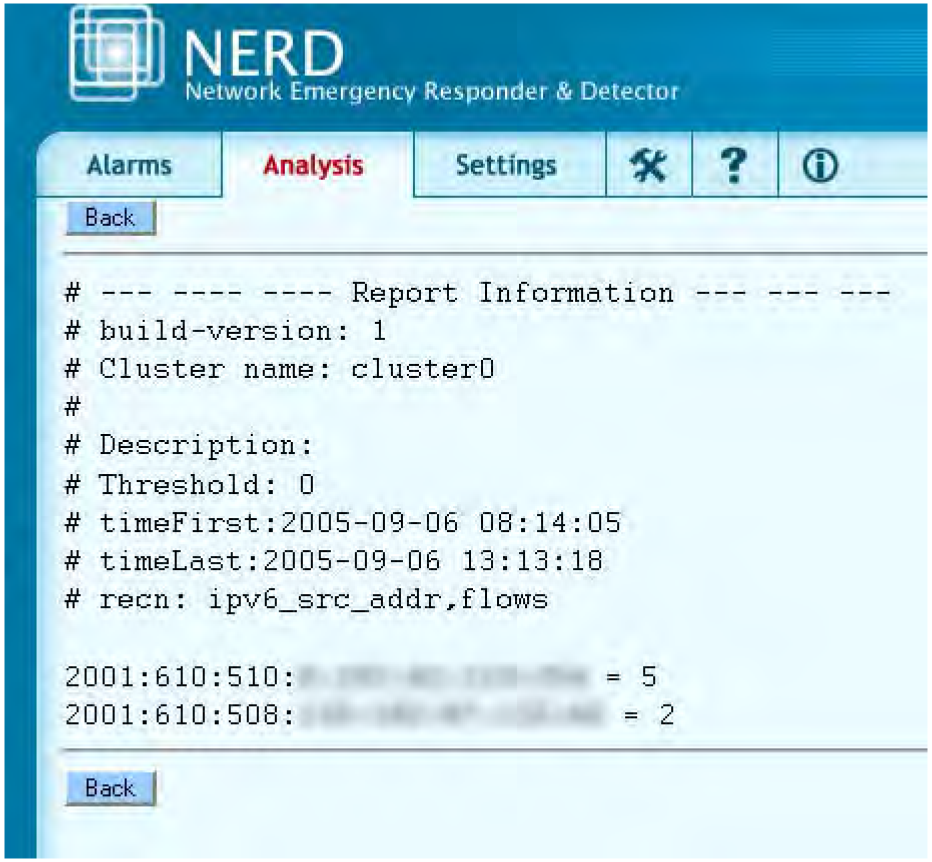
Cluster2 — Del — Add field
src_port X

Cluster3 — Del — Add field
dst_port X

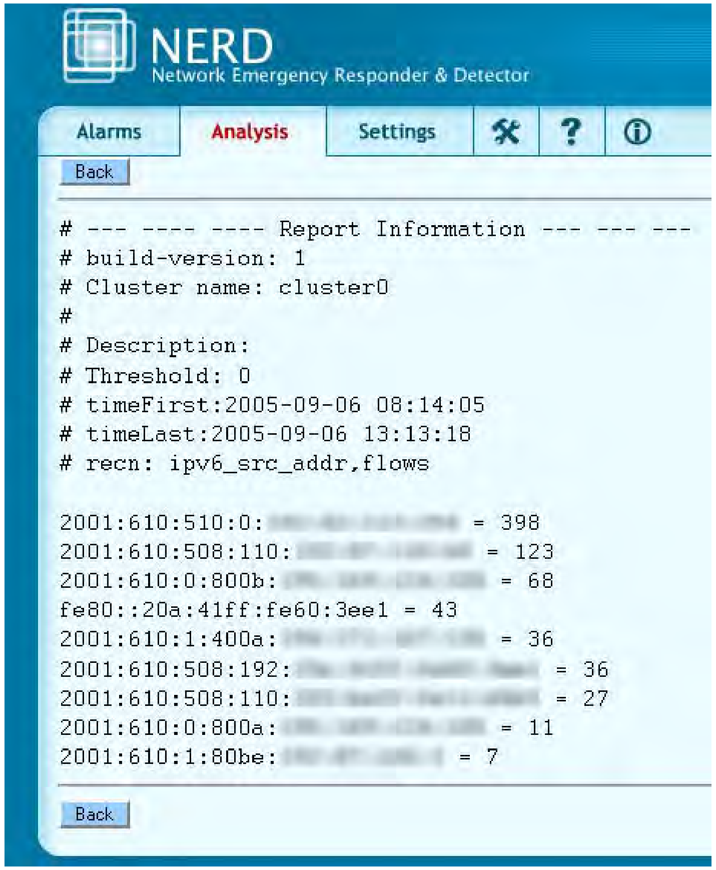
add new cluster

Reset — Analyse

Analysis – IPv6

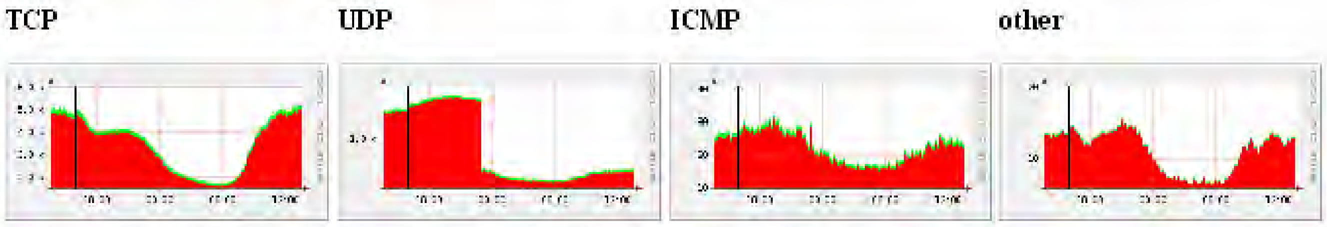


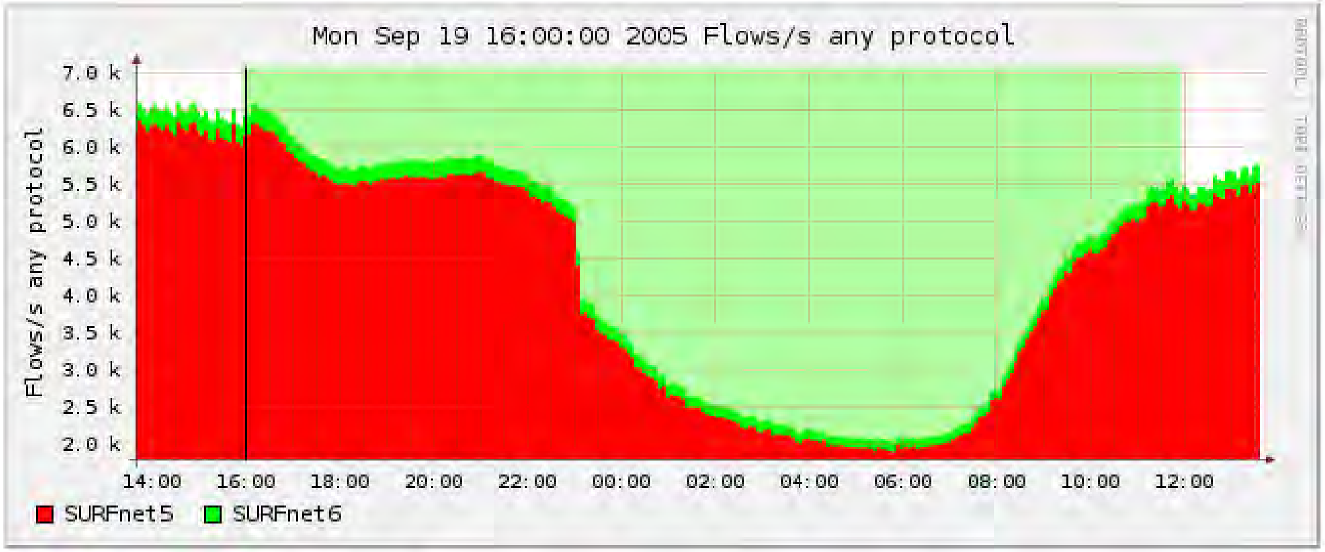
```
# --- ---- ---- Report Information --- --- ---
# build-version: 1
# Cluster name: cluster0
#
# Description:
# Threshold: 0
# timeFirst:2005-09-06 08:14:05
# timeLast:2005-09-06 13:13:18
# recn: ipv6_src_addr,flows

2001:610:510:[redacted] = 5
2001:610:508:[redacted] = 2
```



```
# --- ---- ---- Report Information --- --- ---
# build-version: 1
# Cluster name: cluster0
#
# Description:
# Threshold: 0
# timeFirst:2005-09-06 08:14:05
# timeLast:2005-09-06 13:13:18
# recn: ipv6_src_addr,flows

2001:610:510:0:[redacted] = 398
2001:610:508:110:[redacted] = 123
2001:610:0:800b:[redacted] = 68
fe80::20a:41ff:fe60:3ee1 = 43
2001:610:1:400a:[redacted] = 36
2001:610:508:192:[redacted] = 36
2001:610:508:110:[redacted] = 27
2001:610:0:800a:[redacted] = 11
2001:610:1:80be:[redacted] = 7
```

SURFnet6





Statistics timeslot Sep 19 2005 - 16:00 - Sep 20 2005 - 12:00

Source:	Flows:	Packets:	tcp:	udp:	icmp:	other:	Traffic:	tcp:	udp:	icmp:	other:
✓ SURFnet5	3.8 K/s	19.5 K/s	14.5 K/s	4.6 K/s	29.9 /s	337.4 /s	120.0 Mb/s	89.6 Mb/s	28.8 Mb/s	27.5 Kb/s	1.6 Mb/s
✓ SURFnet6	170.4 /s	663.9 /s	573.6 /s	84.8 /s	2.4 /s	3.1 /s	3.4 Mb/s	3.2 Mb/s	249.4 Kb/s	1.5 Kb/s	22.9 Kb/s

All None

Display: Sum [✓ Rate]

SURFnet

Current Research and Development

- Geant2 JRA2
  - NERD is one of the monitoring toolsets
- LOBSTER project
  - Integration
- Student
  - Analysis and visualisation of worm behaviour
- Ph.D. from Vrije Universiteit (VU)
  - Interaction of Netflow and Full Packet inspection
- From application to framework
  - Other data sources, combining different data
  - Other data output

Questions

- More information and download of NERD
  - www.nerdd.org

IP Flow Information Export (IPFIX): Applicability and Future Suggestions for Network Security

Elisa Boschi, Tanja Zseby, Lutz Mark, Thomas Hirsch
Fraunhofer FOKUS,
Berlin, Germany
{boschi, zseby, mark, hirsch}@fokus.fraunhofer.de

Abstract

This year, the IP Flow Information Export (IPFIX) protocol will become standard for exporting flow information from routers and probes. Standardized methods for packet selection and the export of per packet information will follow soon from the IETF group on packet sampling (PSAMP). The future availability of network information in a standardized form enables a wide range of critical applications for Internet operation including, accounting, QoS auditing and detection of network attacks. In this paper we present the IPFIX protocol, and discuss its applicability with a special focus on network security. We propose a coupling of IPIFX with AAA functions to improve the detection and defense against network security incidents and for Inter-domain information exchange based on IPIFX utilizing secure transmission channels provided by the AAA architecture.

1. Introduction

The IP Flow Information eXport (IPFIX) protocol defines a format and protocol for the export of flow information from routers or measurement probes. In the past a lot of proprietary solutions were developed for flow information export (e.g. Cisco NetFlow, InMon sFlow, NeTraMet, etc). Now after several years of lively discussions the IETF is about to submit a standard for flow information export, the IPFIX protocol.

Capturing flow information plays an important role for network security, both for detection of security violation, and for subsequent defence. Attack and intrusion detection is one of the five target applications that require flow measurements on which the requirements definition has been based: usage-based accounting, traffic analysis, traffic engineering, QoS monitoring and intrusion detection (Cf. [RFC 3917]).

In this paper we summarize the IPFIX protocol, describe our implementation of it and discuss its applicability for a number of different applications. Particularly, we analyze the IPFIX applicability to security related applications such as anomaly and intrusion detection and discuss the coupling of IPFIX with AAA in the context of inter-domain information exchange.

2. IP Flow Information Export

2.1 IPFIX Summary

Based on the requirements defined in [RFC3917] different existing protocols (NetFlow v9, Diameter, CRANE, IPDR) where evaluated as candidates to provide the basis for a future IPFIX protocol [RFC3955]. The result of the evaluation was that NetFlow v9 provides the best basis for it.

IPFIX is a general data transport protocol that is easily extensible to suit the needs of different applications. The protocol is flexible in both flow key and flow export. The Flow Key defines the properties used to select flows and can be defined depending on the application needs. Flow information is exported using flow data records and the information contained in those records can be defined using template records. A template ID uniquely identifies each template record and provides the binding between template and data records.

As requested by the IESG, IPFIX transport has to fulfil certain reliability and security requirements. Therefore PR-SCTP has been chosen as mandatory basic transport protocol for IPFIX for all compliant implementations. TCP and UDP can be used as optional protocols. Preference to PR-SCTP was given because it is congestion-aware and reduces bandwidth in case of congestion but still has a much simpler state machine than TCP. This saves resources on lightweight probes and router line cards.

2.2 FOKUS IPFIX Implementation

Fraunhofer FOKUS has developed and uses an Internet measurement application for distributed IP traffic and quality of service measurements. The software is called OpenIMP and consists of a central measurement controller and multiple distributed probes. The probes are remote controlled and send their measurement data to dedicated collectors. The measurement result transfer was implemented via files that were generated on the probes and were sent to the collector via ftp or scp. With the definition of IPFIX there appeared a smart and powerful alternative to the file transfer. So we decided to integrate IPFIX into the measurement application. The implementation was started before the definition of the IPFIX protocol has been finished. This offered us the possibility to send comments to the working group and to influence the further definition of the IPFIX protocol.

The IPFIX protocol was implemented within a separate C-library. This has the advantage that the implementation can be used outside OpenIMP. Using C makes it easy to integrate the code into e.g. C++ or JAVA applications and has the advantage of leading to small binaries. The library supports the IPFIX exporting and collecting processes via providing functions to export and to collect data using IPFIX data types and protocol.

The IPFIX protocol is not complex, which keeps the implementation simple. IPFIX messages can be transferred using SCTP, TCP or UDP as bearer protocol. An IPFIX implementation has to support SCTP-PR whereas support for TCP and UDP is optional. Unfortunately currently there is no major operating system with full support for SCTP-PR (at least the authors are not aware of one). So at present an application has to use TCP to enable the export over ipv6 networks. To test the SCTP code we used Debian Linux with kernel 2.6 that has support for SCTP-PR over ipv4.

The main task of the IPFIX exporter is to take the measurement data from one or more metering processes and to send the IPFIX messages to the data collectors. The exporter has to take care that the templates are sent prior to the related data records. For SCTP and TCP the templates have to be resent on a connection reestablishment. For UDP templates have to be resent after a configured timeout. This makes the implementation a bit more complex and requires the exporting process to store all active template definitions.

The IPFIX collector has to maintain a list of sources and per source a list of templates to decode incoming data templates. Because of the template feature of IPFIX the collector does not need any knowledge of the transferred data. All information needed to decode all kind of data is transferred via template records.

2.3 Applicability Scenarios: Accounting and QoS Monitoring

Usage-based accounting is one of the major applications for which the IPFIX protocol has been developed. IPFIX data records provide fine-grained measurement results for highly flexible and detailed resource usage accounting (i.e. the number of transferred packets and bytes per flow). Internet Service Providers (ISPs) can use this information to migrate from single fee, flat-rate billing to more flexible charging mechanisms based on time of day, bandwidth usage, application usage, quality of service, etc.

In order to realize usage-based accounting with IPFIX the flow definition has to be chosen in accordance with the tariff model. If for example the tariff is based on individual end-to-end flows, accounting can be realized with a flow definition determined by source address, destination address, protocol, and port numbers. Another example is a class-dependent tariff (e.g. in a DiffServ network); in this case, flows can be distinguished just by the DiffServ codepoint (DSCP) and source IP address.

QoS monitoring is the passive observation of transmission quality for single flows or traffic aggregates in the network and is one of the target applications of IPFIX. One example of its usefulness is the validation of QoS guarantees in service level agreements (SLAs). IPFIX data records enable ISPs to perform a detailed, time-based, and application-based usage analysis of a network.

3. IPFIX Applicability and Future Suggestions for Detection of and Defense against Network Attacks

The applications described in this section are related to the detection and report of intrusions and anomalous traffic. While describing the applicability of the protocol, we discuss how IPFIX could further support detection of, and reaction to, network attacks.

3.1 Packet Selection and Packet Information Export

For some scenarios, the detection of malicious traffic may require further insight into packet content. The PSAMP working group works on the standardization of packet selection methods [ZsMD05] and the export of per packet information [Duff05], [PoMB05]. Recently, the IETF PSAMP [PSAMP] group has decided to also use the IPFIX protocol for the export of per packet information. That means, in future we will get also per packet information from routers in a standardized way.

3.2 Detection of network incidents and malicious traffic

IPFIX provides useful input data for basic attack detection functions such as reporting unusually high loads, number of flows, number of packets of a specific type, etc. It can provide details on source and destination addresses, along with the start time of flows, TCP flags, application ports and flow volume. This data can be used to analyze network security incidents and identify attacks like DoS attacks, worm propagation or port scanning. Further data analysis and post-processing functions may be needed to generate the metric of interest for specific attack types.

Already basic IPFIX information allows detecting common attack schemes: A distributed DoS attack generates a large number of flows, often with a high data volume. The *number of newly detected source addresses* is commonly used [TaHo04] as a metric for detecting distributed activities. It correlates strongly with the flow count metric of IPFIX. Also, sudden increases in the occurrence of unusual IP or TCP flags (e.g. “Don’t Fragment”) can be an indicator for malicious traffic [TaAl02, SiPa04]. Based on the IPFIX information, derived metrics can highlight changes and anomalies. The most successful methods for anomaly detection to date are non-parametric change point detection algorithms, such as the cumulative sum (CUSUM) algorithm [WaZS02]. The integration of previous measurement results helps to assess traffic changes over time for detection of traffic anomalies. A combination with results from other observation points allows assessing the propagation of the attack and can help locate the source of an attack.

Detecting security incidents in real-time would require the pre-processing of data already at the measurement device and immediate data export in case a possible incident has been identified. This means that IPFIX reports must be generated upon incident detection events and not only upon flow end or fixed time intervals. IPFIX works in push mode. That means data records are automatically exported without waiting for a request. Placing the responsibility for initiating a data export at the exporting process is quite useful for detection of security incidents. The exporting process can immediately trigger the export of information if suspicious events are observed (e.g. sudden increase of the number of flows).

Security incidents could become a threat to IPFIX processes themselves. If an attack generates a large amount of flows (e.g. by sending packets with spoofed addresses or simulating flow termination), exporting and collecting process may get overloaded by the immense amount of data records that are exported. A flexible deployment of packet or flow sampling methods can prevent the exhaustion of resources.

3.3 Sharing Information with Neighbor Domains

For inter-domain measurements it is required to exchange *result* data, and eventually to allow remote configuration, across multiple administrative domains. Result data can be both measurements of QoS metrics on an end-to-end path, or monitoring information for troubleshooting, or information regarding attacks (either notifications of anomalous traffic or specific measurements to get further insight in case suspicious behavior was observed). Although ISPs can control and monitor their own network, they have minimal or no information at all about the characteristics and performance of other networks, nor the means of requesting and acquiring it. IPFIX provides the standard format and protocol for this information exchange.

3.4 Sharing Information with AAA Functions

One approach to do this is to use existing AAA components which provide a secure data transfer between domains by using the DIAMETER protocol and also provide functions for authentication and authorization which are useful to control access to data [RFC3334]. Furthermore, AAA servers usually keep accounting and auditing requirements, which can be used to directly derive measurement demands. For anomaly and intrusion detection the strong relation to AAA components can provide further benefits. Potential attackers can be identified and stopped from injecting traffic into the network. This is especially powerful, if AAA components from different administrative domains work together.

The combination of IPFIX and AAA functions can be beneficial also for attack detection. Such an interoperation enables further means of attack detection, advanced defense strategies and secure inter-domain cooperation. A AAA system has secure channels to neighbor AAA servers and can inform neighbors about incidents or suspicious observations. Through this system an ISP could also react to an attack by for example requesting the denial of access for potential attackers. A further benefit would be if AAA functions could invoke further measurements.

4. Conclusions

IPFIX is the upcoming standard for IP flow information export. The protocol is well suited to support applications such as QoS measurement, accounting, and anomaly and intrusion detection. In particular, for security applications, IPFIX can be used to exchange information with neighbors not only about incidents or anomalous traffic or to receive information previously requested to track attackers or classify the attacks. The joint use of IPFIX and AAA functions can add further benefits and be useful to track and stop attackers.

Some implementations of the protocol already exist, one of them coming from the authors of this paper, and interoperability events have been planned in the next months.

5. References

[RFC3917] J. Quittek, T. Zseby, B. Claise, S. Zander, Requirements for IP Flow Information Export, October 2004

[Clai05] B. Claise (Editor), IPFIX Protocol Specification, Internet Draft <draft-ietf-ipfix-protocol-14.txt>, work in progress, May 2005

[PoMB05] G. Pohl, L. Mark, E. Boschi Export of per-packet information using IPFIX, Internet Draft <draft-pohl-perpktinfo-03.txt>, work in progress, February 2005

[QuBr05] J. Quittek, S. Bryant, J. Meyer, Information Model for IP Flow Information Export, Internet Draft <draft-ietf-ipfix-info-06>, work in progress, February 2005

[PSAMP] http://www.ietf.org/html.charters/psamp-charter.html

[Duff05] Nick Duffield (Ed.), A Framework for Packet Selection and Reporting, Internet Draft draft-ietf-psamp-framework-08, work in progress, January 2005

[RFC3334] T. Zseby, S. Zander, G. Carle, Policy-Based Accounting, Request for Comments: 3334, October 2002

[TaHo04] Application and Analyses of Cumulative Sum to Detect Highly Distributed Denial of Service Attacks using Different Attack Traffic Patterns", H. Takada and U. Hofmann, IST INTERMON newsletter issue 7, Feb 2004

[ZsMD05] Tanja Zseby, Maurizio Molina, Nick Duffield, Saverio Niccolini, Fredric Raspall: Sampling and Filtering Techniques for IP Packet Selection, Internet Draft <draft-ietf-psamp-sample-tech-06.txt>, work in progress, February 2005

[ZsBB05] Tanja Zseby, Elisa Boschi, Nevil Brownlee, Benoit Claise, IPFIX Applicability, Internet Draft <draft-ietf-ipfix-as-05.txt>, work in progress, May 2005

[TaAl02] An Empirical Analysis of NATE Network Analysis of Anomalous Traffic Events. Carol Taylor, Jim Alves-Foss, CS Department, U Idaho

[SiPa04] V. Siris & F. Papagalou; Application of Anomaly Detection Algorithms for Detecting SYN Flooding Attacks; Proceedings of IEEE Global Telecommunications Conference (Globecom 2004), Dallas, USA, 29 November - 3 December 2004.

[WaZS02] H Wang, D Zhang, K G Shin, Detecing SYN flooding attacks, in Proc IEEE INFOCOM, 2002

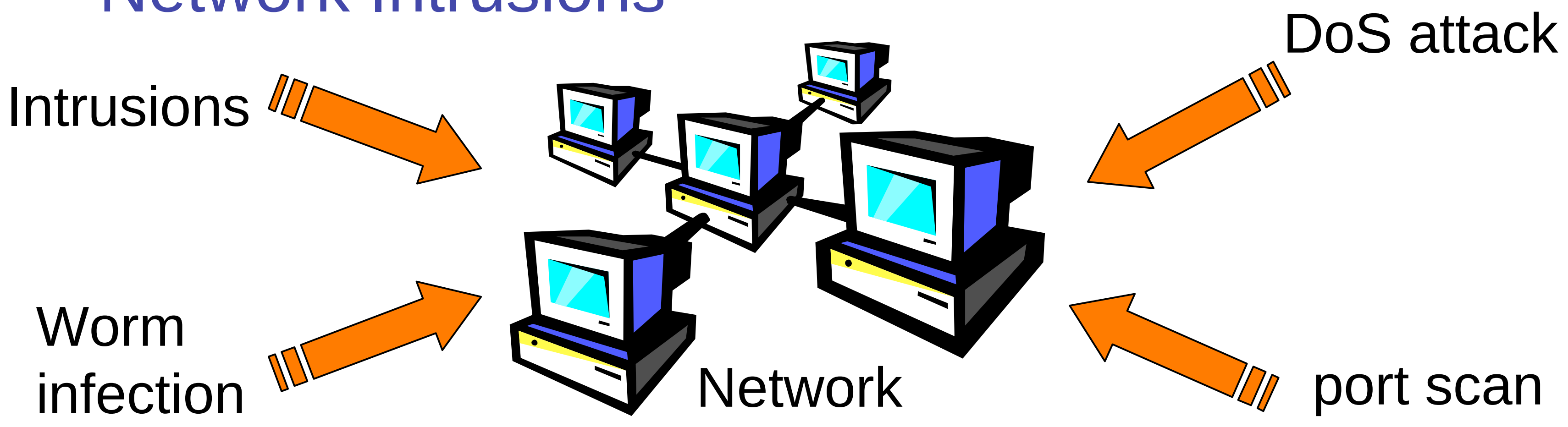
VisFlowConnect-IP: An Animated Link Analysis Tool For Visualizing Netflows

Xiaoxin Yin*, William Yurcik, Adam Slagell

SIFT Research Group
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign

Motivations

- Network Intrusions

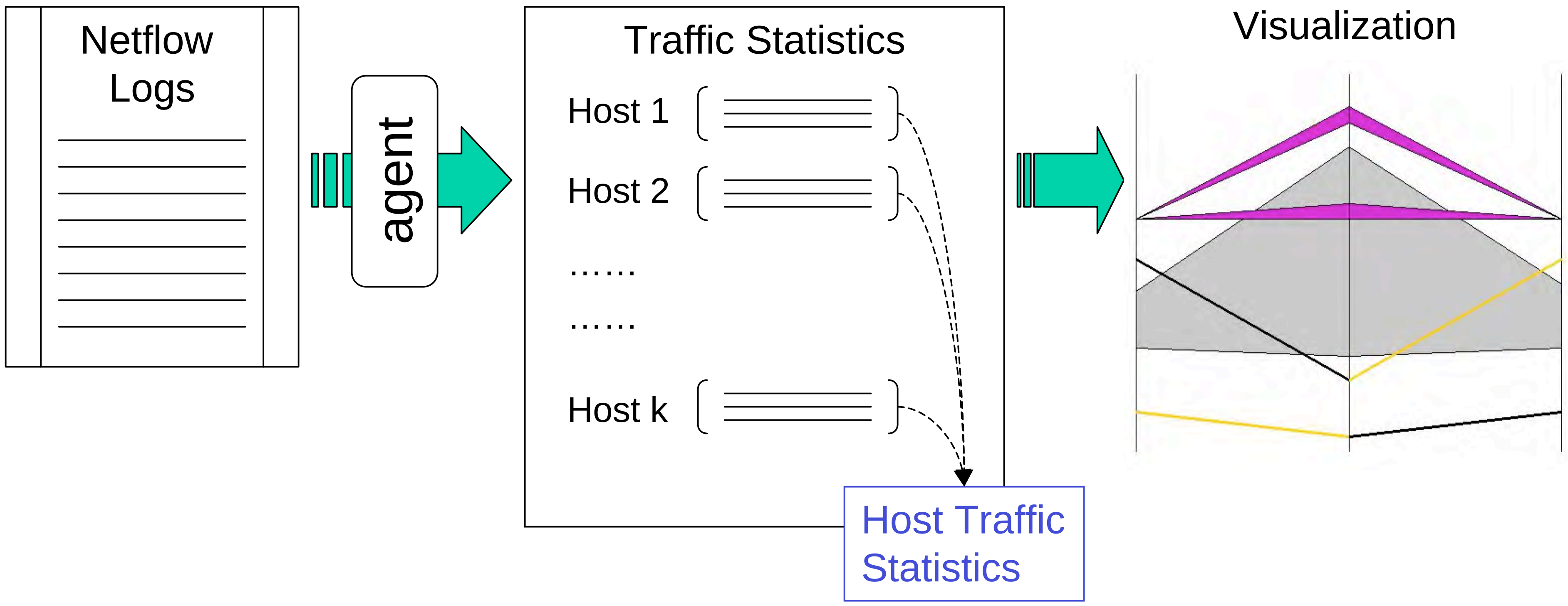


- Intrusion Detection Systems
  – Misuse detection: find signatures of intrusions
  – Anomaly detection: model normal behaviors
- Visualize network traffic
  – So that intrusions are apparent to human

Overview

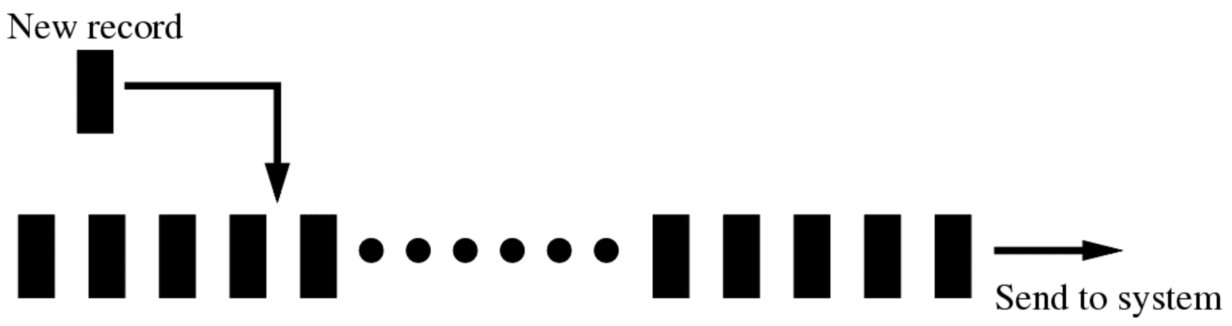
- Visualizing network traffic as a graph
  - Hosts → nodes in graph
  - Traffic → flow in graph
  - other conceptual models are possible (i.e. NVisionIP)
- Visualizing by ***animation***
  - Show network dynamics by animation
  - Visualize traffic within a user adjustable time window
- High scalability
  - Run continuously for long periods
  - Uses constant storage to process large data sets or high speed streaming data.

VisFlowConnect System Design



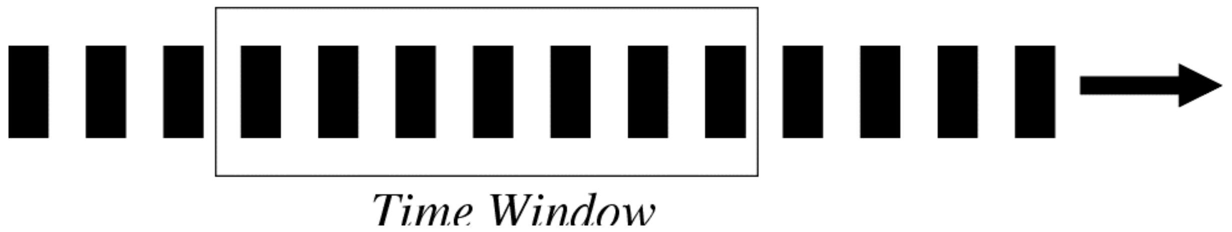
Reading Netflow Logs

- An agent reads records from file or in real time
  - Send a record to VisFlowConnect when requested
- Reorder Netflow records with record buffer
  - Records are not strictly sorted by time stamps
  - Use a record buffer



Time Window

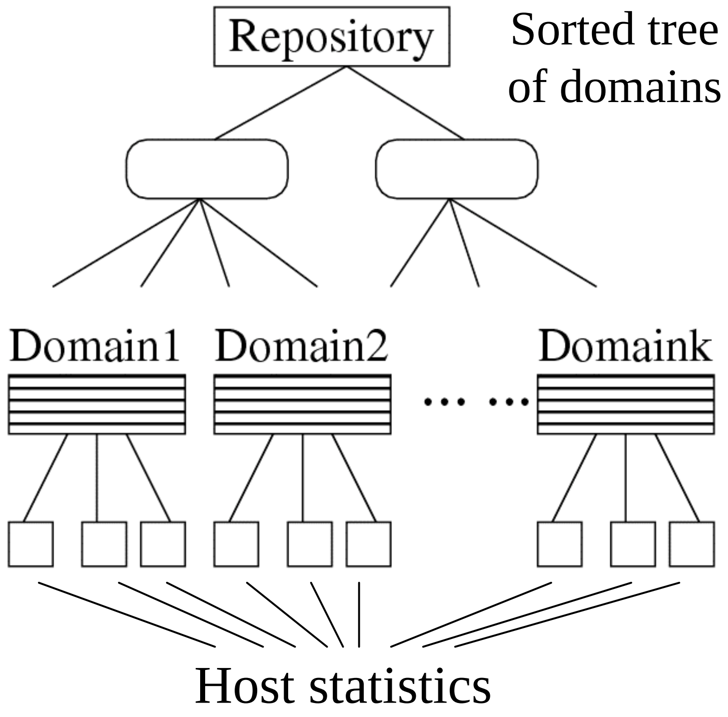
- User is usually interested in most recent traffic (e.g., in last minute or last hour)
- VisFlowConnect only visualizes traffic in a user adjustable time window



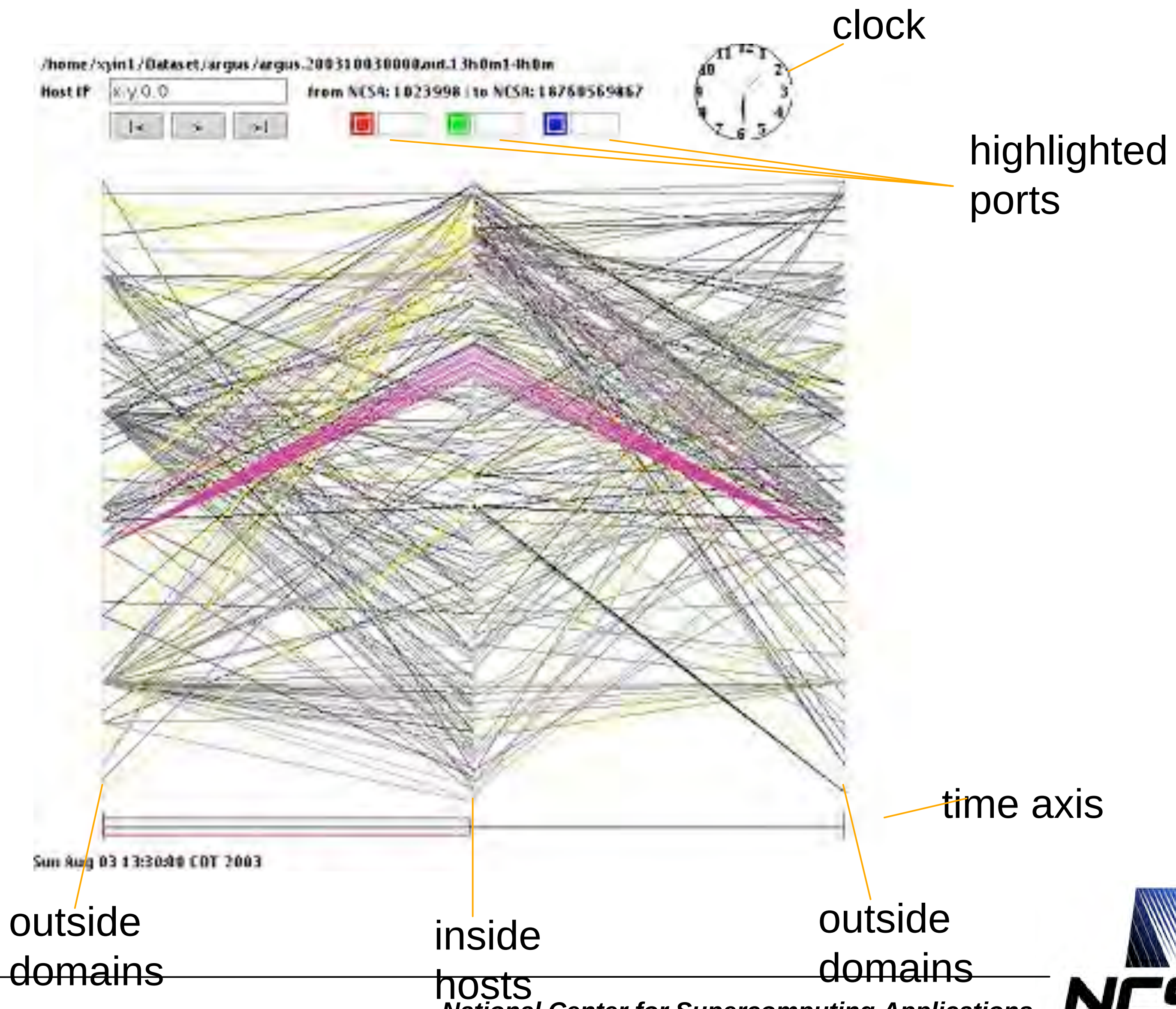
  – Update traffic statistics when
    - A record comes into time window
    - A record goes out of time window

Storing Traffic Statistics

- Store traffic statistics involving each domain by a sorted tree
  - Only necessary information for visualization is stored
  - Statistics for every domain or host can be updated efficiently



VisFlowConnect External View

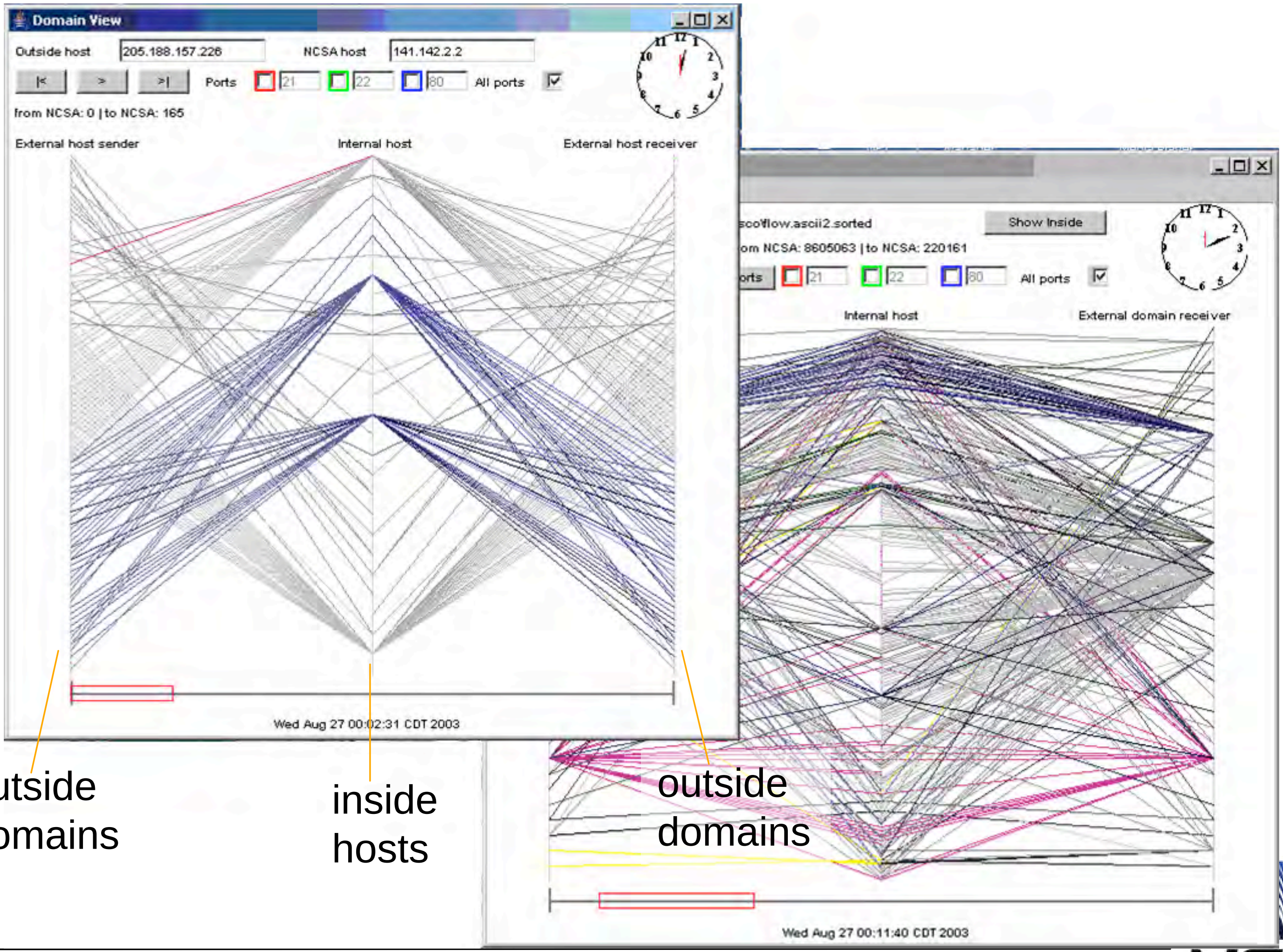

/home/xyin1/Dataset/argus/argus.200310030000ard.13h0m14h0m

Host IP x.y.0.0

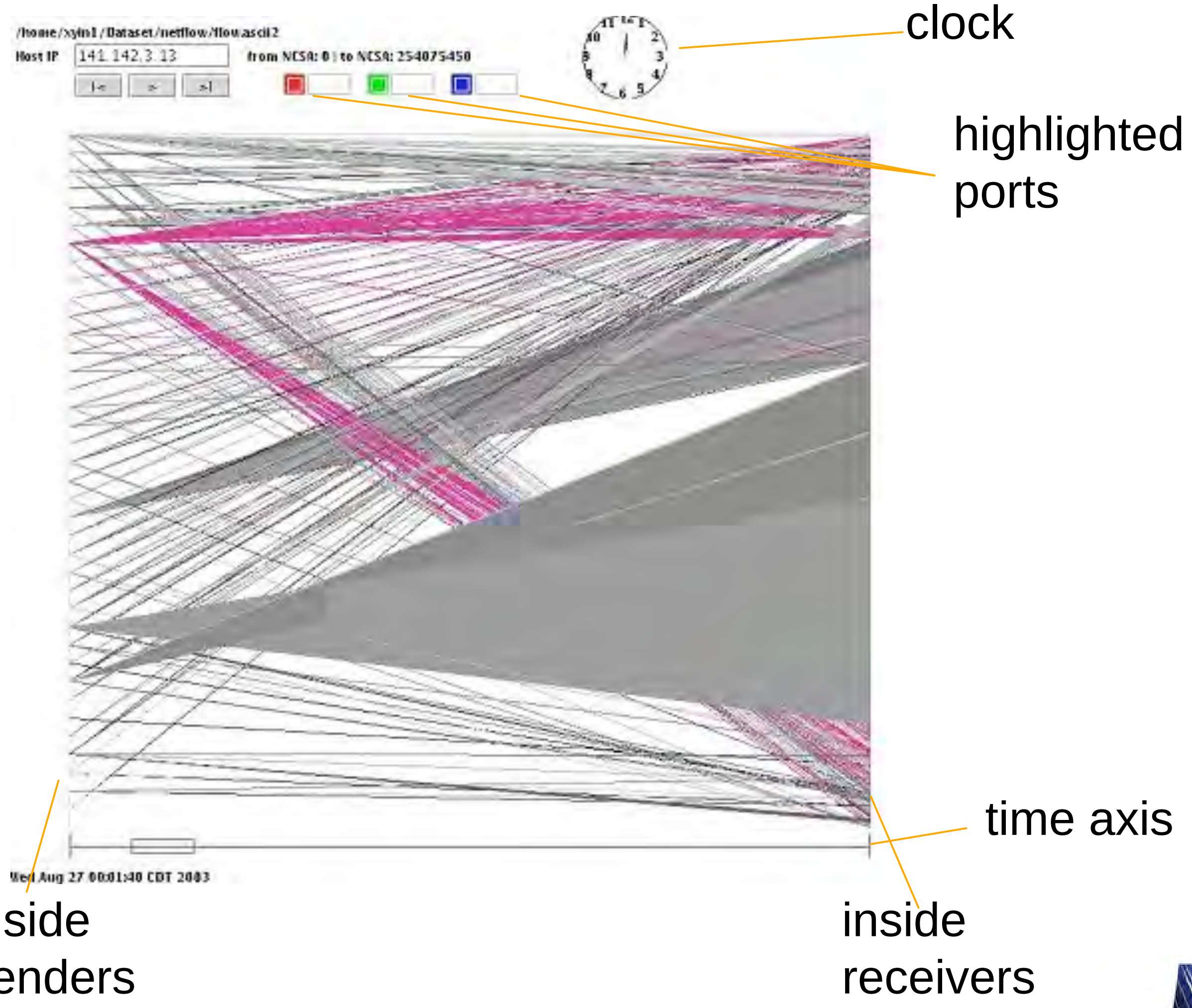
from NCSA: 1023998 | to NCSA: 18760569867

Sun Aug 03 13:30:00 CDT 2003

VisFlowConnect Domain View



VisFlowConnect Internal View



/home/xyin1/Dataset/netflow/flow.ascii2

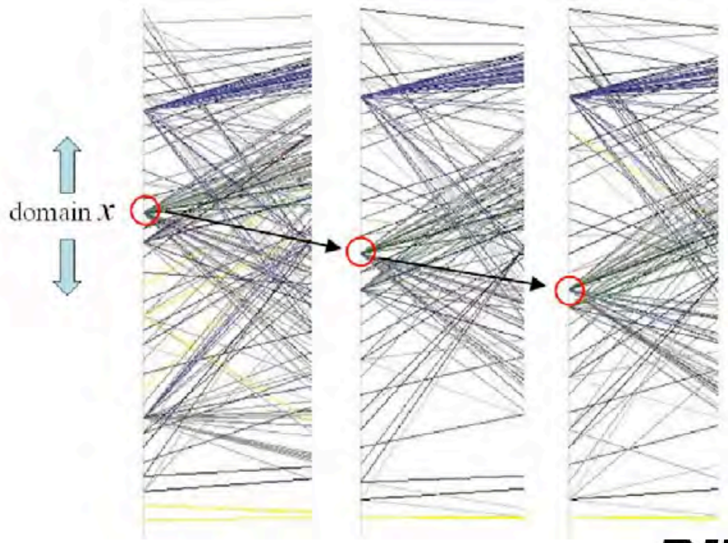
Host IP 141.142.3.13

from NCSA: 0 | to NCSA: 254075450

Wed Aug 27 00:01:40 CDT 2003

Creating Animation

- Visualizing traffic statistics with time
  - Update visualization after each time unit
- How to arrange domains/hosts?
  - Only hundreds of domains/hosts can be put on one axis
  - Domains/hosts may be added or removed with time
  - The position of each domain/host must be fairly stable
- Solution: sort domains/hosts by IP
  - Each domain/IP may move up or down



Filtering Capability

- Filter out regular traffic
  - E.g., DNS traffic, common HTTP traffic
- Work like a spam-mail filter
  - User specifies a list of filters:

    +: (SrcIP=141.142.0.0−141.142.255.255), (SrcPort=1−1000)

    //keep all records from domain 141.142.x.x, from port 1 – 1000

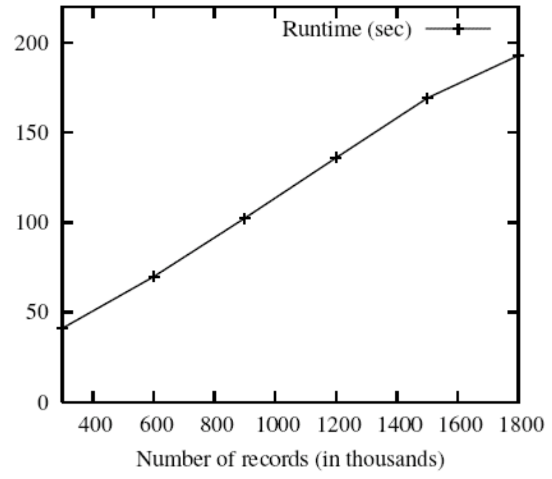
    −: (SrcPort=80)

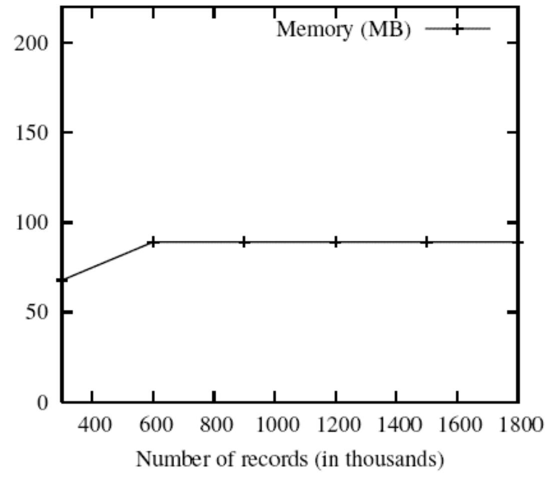
    −: (DstPort=80)

    //discard records of http traffic
  - Each record is passed through each filter
  - Last match is used to decide whether keep this record or not

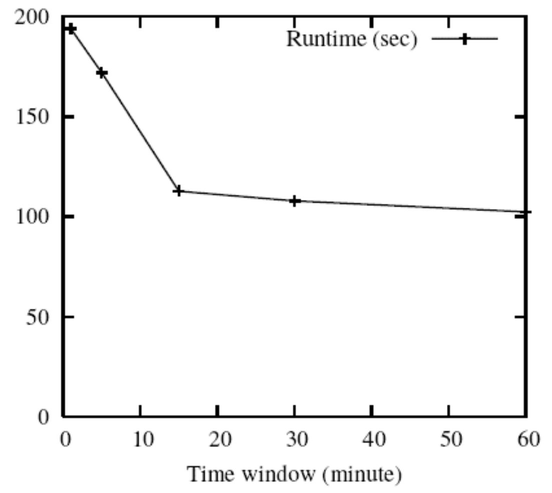
Scalability Experiments

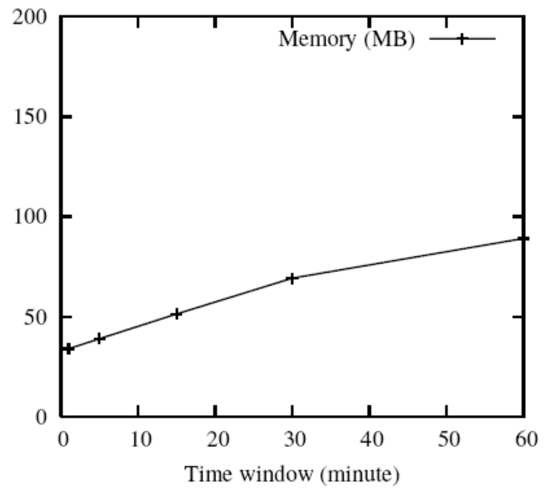
Runtime and memory w.r.t. number of records





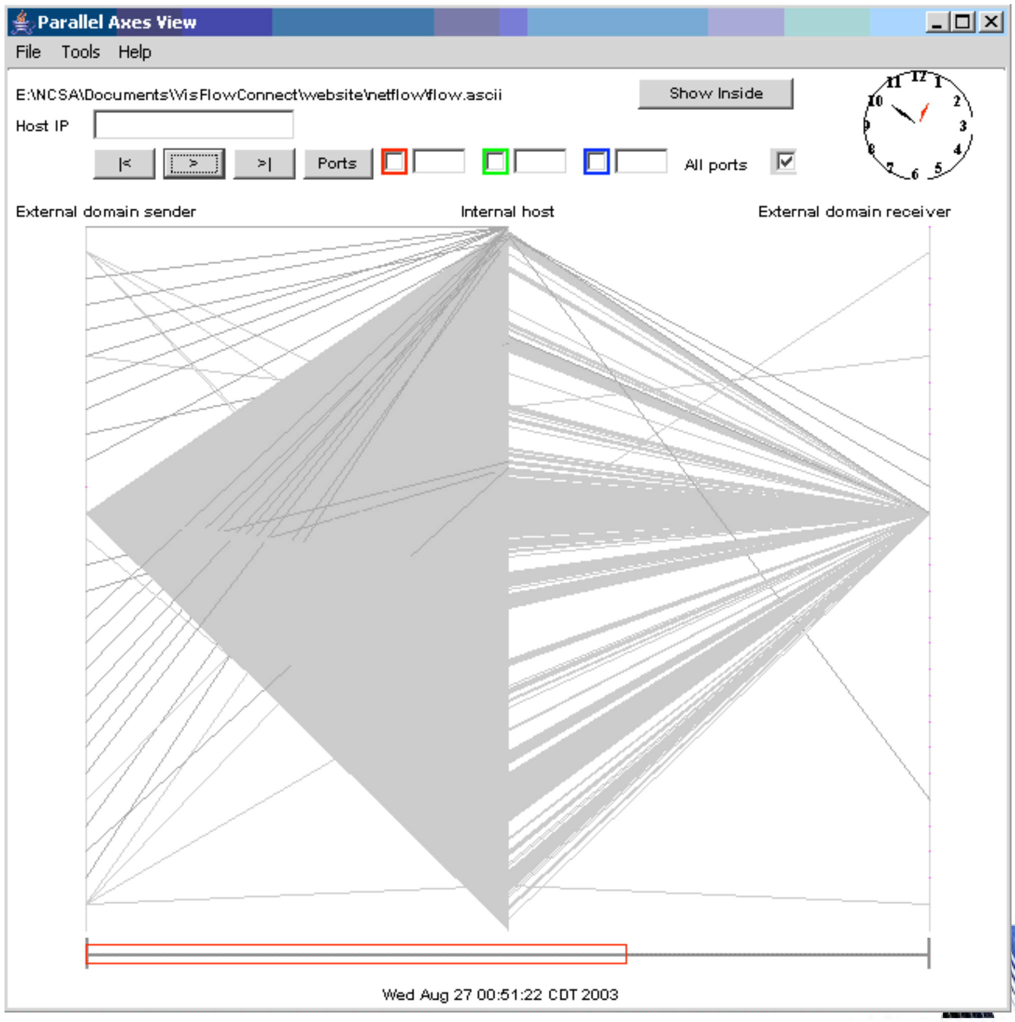
Runtime and memory w.r.t. size of time window





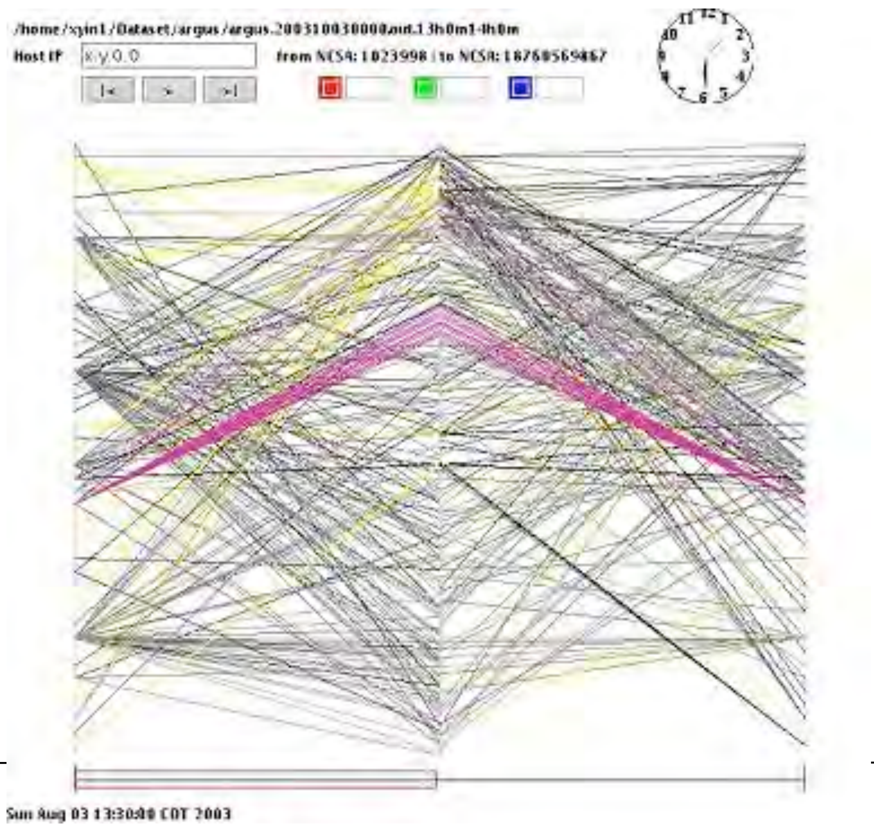
Example 1: MS Blaster Virus

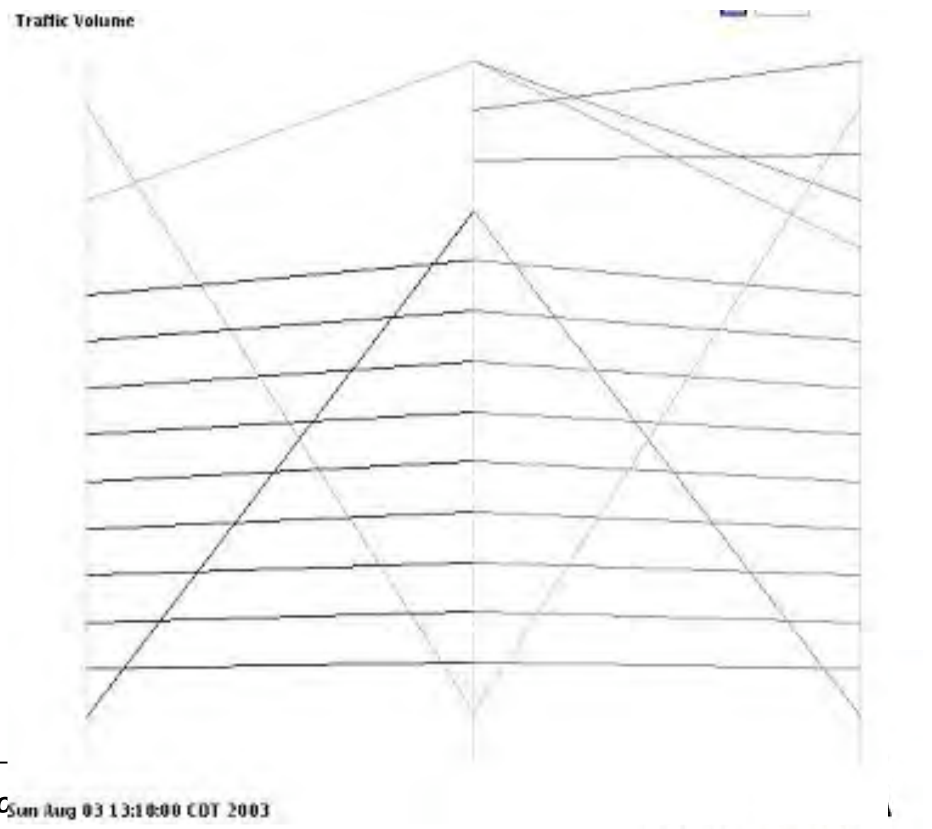
- MS Blaster virus causes machines to send out packets of size 92 to many machines



Example 2: 1-to-1 Communication of Clusters

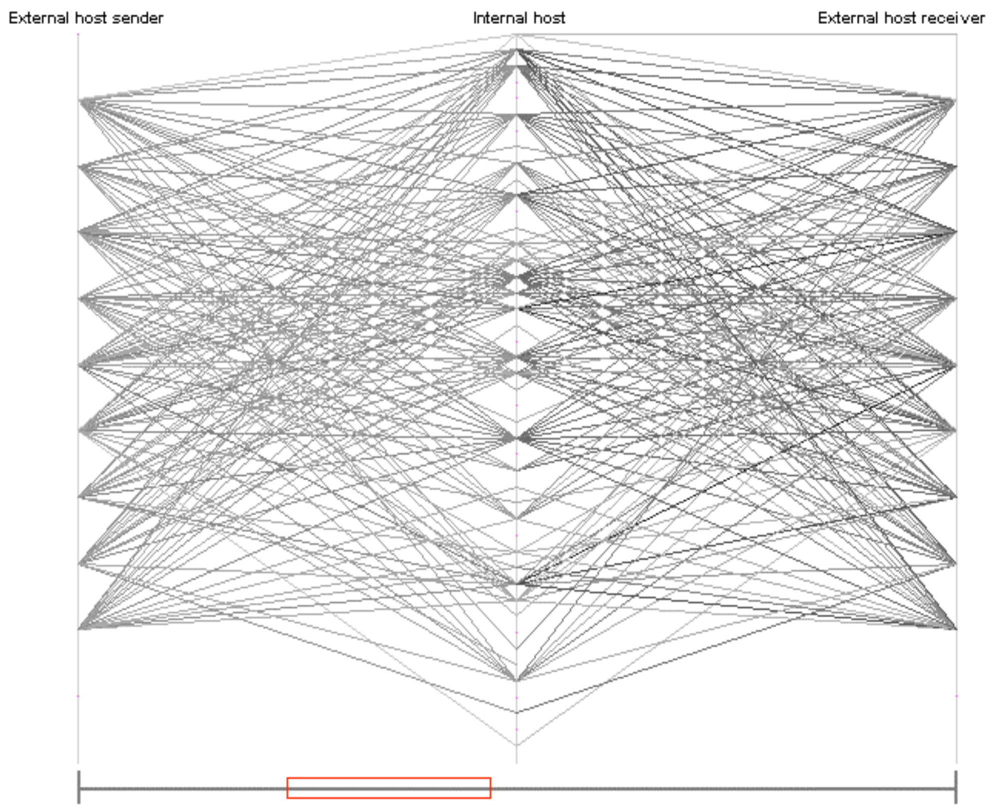
- We found there are two sets of hosts of continuous IPs have 1-to-1 communications with each other. Finally we found they are two clusters.


/home/xyin1/Dataset/argus/argus.200310030000and.13h0m14h0m
Host IP
x.y.0.0
from NCSA: 1023998 | to NCSA: 18760569467
|<
>
>|
Sun Aug 03 13:30:00 CDT 2003



Example 3: Web Crawlers

- We found 9 hosts in a domain connecting to many http servers in our network
  - Their IPs are from Google.com: Web crawling



More Information

- VisFlowConnect is being ported to other specialized security domains
  - Storage security (two publications pending)
  - Cluster security
- Distribution Website
  - http://security.ncsa.uiuc.edu/distribution/VisFlowConnectDownLoad.html

    VisFlowConnect are downloadable there
- Publications of SIFT Group
  - http://www.ncassr.org/projects/sift/papers/

Thank You!

Xiaoxin Yin

<xyin1@uiuc.edu>

NCSA SIFT Group

University of Illinois

Covert Channel Detection Using Process Query Systems

➡ **Annarita Giani**
Vincent Berk
George Cybenko

Institute for Security Technology Studies
Thayer School of Engineering
Dartmouth College
Hanover, NH

FLoCon 2005

MOTIVATION

CNN.COM
Sunday, June 19, 2005 Posted: 0238 GMT (1038 HKT)

NEW YORK -- The names, banks and account numbers of up to 40 million credit card holders may have been accessed by an unauthorized user, MasterCard International Inc. said.

Interest in network and computer security

Started investigating DATA EXFILTRATION

COVERT CHANNELS are the most subtle way of moving data. They easily bypass current security tools.

Until now there has not been enough interest. So detection is still at the first stage.

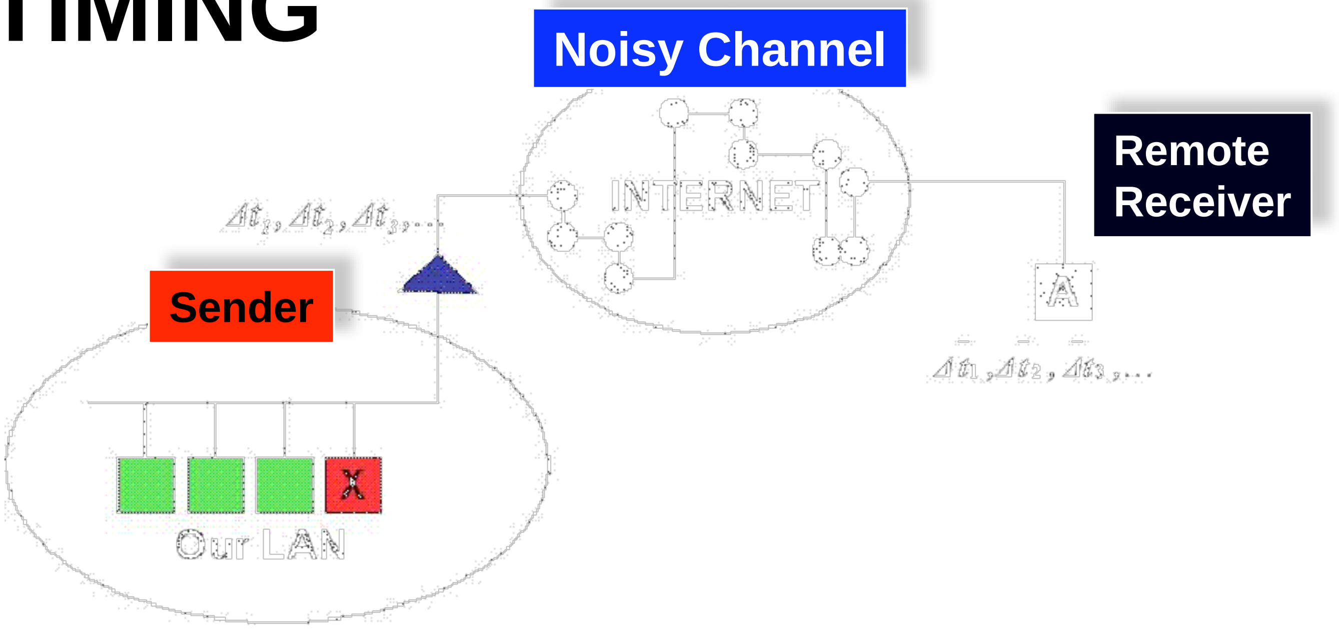
OUTLINE

- ➡ Covert Channels
- Process Query Systems
- Detection of covert channels using a PQS

“A communication channel is covert if it is neither designed nor intended to transfer information at all.” (Lampson 1973)

“Covert channels are those that use entities not normally viewed as data objects to transfer information from one subject to another.” (kemmerer 1983)

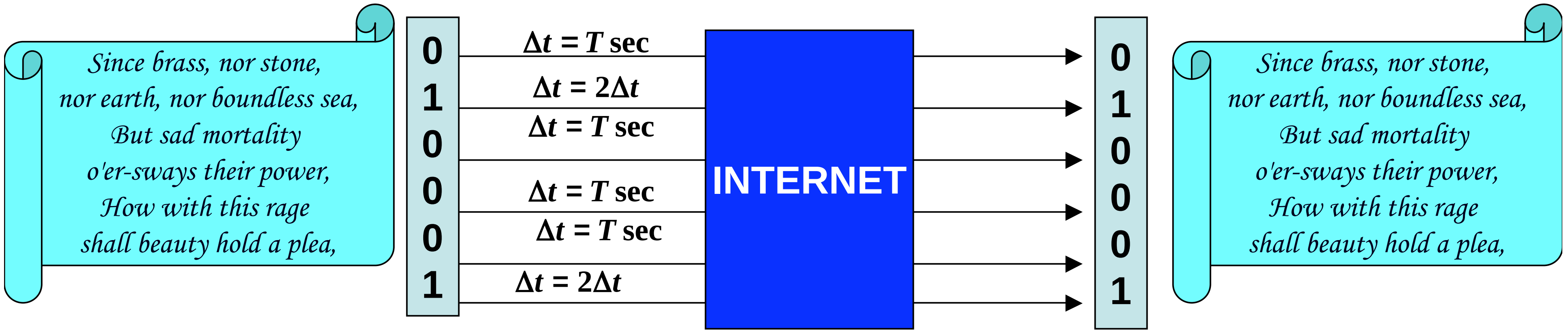
EXAMPLE: TIMING COVERT CHANNEL



Noisy Channel
Remote Receiver
INTERNET
$\Delta t_1, \Delta t_2, \Delta t_3, \ldots$
Sender
Our LAN
$\Delta t_1, \Delta t_2, \Delta t_3, \ldots$

Two approaches

1. Information Theory
2. Statistical analysis

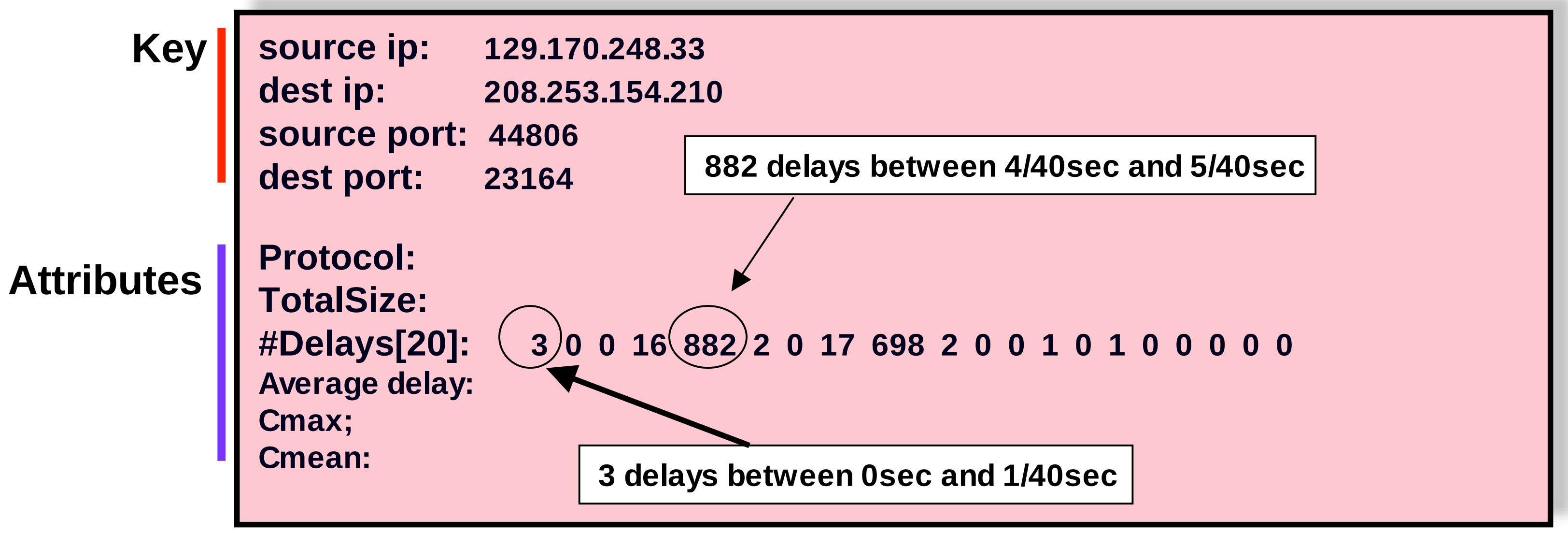


Sensor

Traffic is separated in connection types

We built a package that registers the time delays between consecutive packets for every network traffic flow.

Given an interval of time we build the following node:



Key

source ip: 129.170.248.33
dest ip: 208.253.154.210
source port: 44806
dest port: 23164

Attributes

Protocol:
TotalSize:
#Delays[20]: 3 0 0 16 882 2 0 17 698 2 0 0 1 0 1 0 0 0 0 0
Average delay:
Cmax;
Cmean:

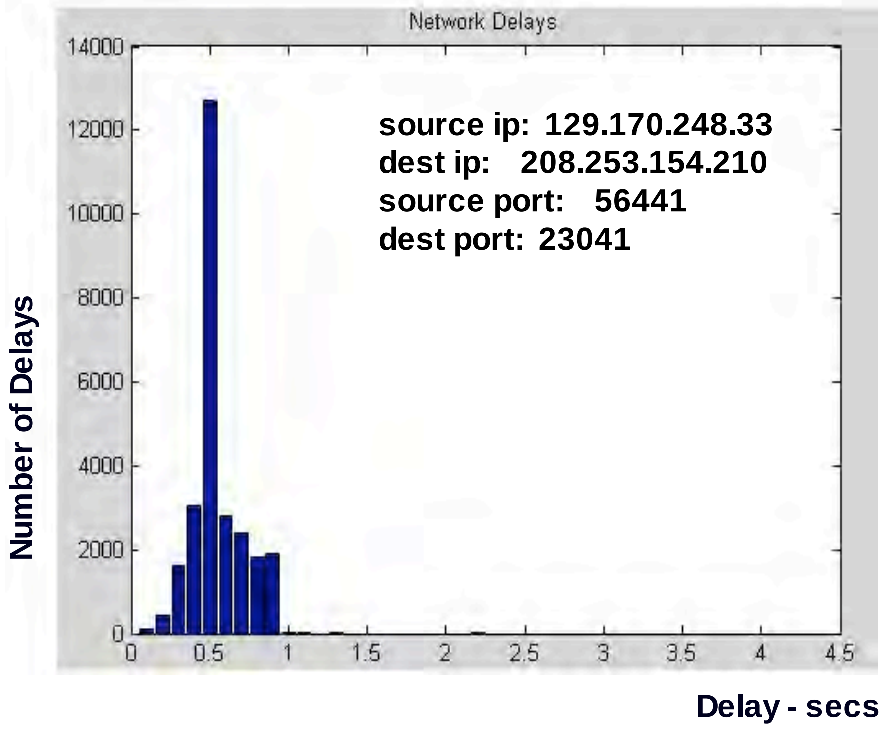
882 delays between 4/40sec and 5/40sec

3 delays between 0sec and 1/40sec

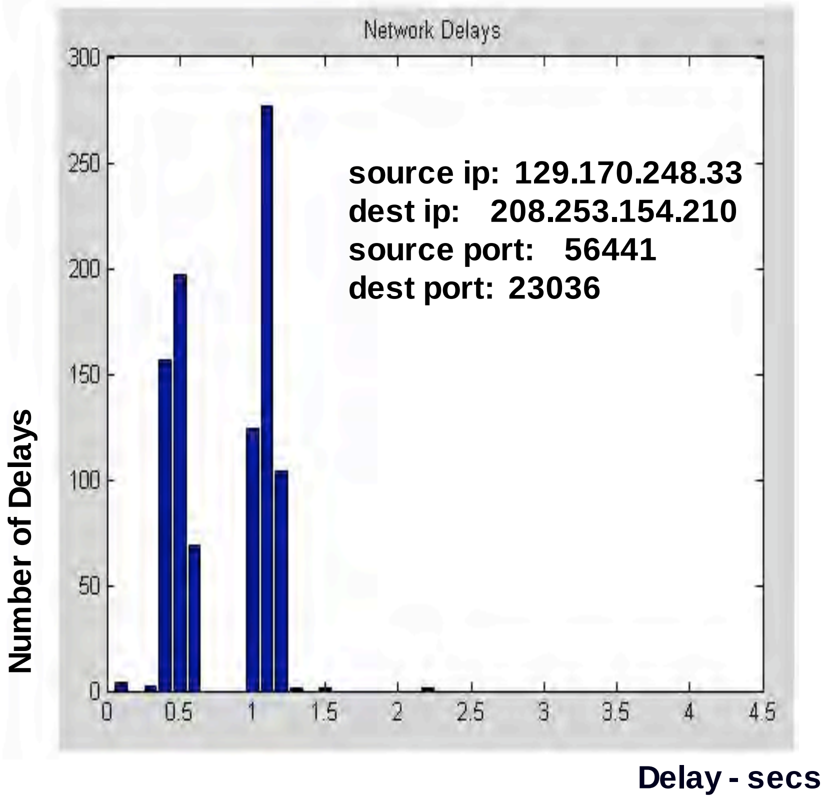
Covert Channels

Assumptions of the experiments:

- **No malicious noise.**
- **Binary source.**



source ip: 129.170.248.33
dest ip: 208.253.154.210
source port: 56441
dest port: 23041



source ip: 129.170.248.33
dest ip: 208.253.154.210
source port: 56441
dest port: 23036

OUTLINE

- Covert Channels
- ➡ Process Query Systems
- Detection of covert channels using a PQS

Process Query Systems for Homeland Security

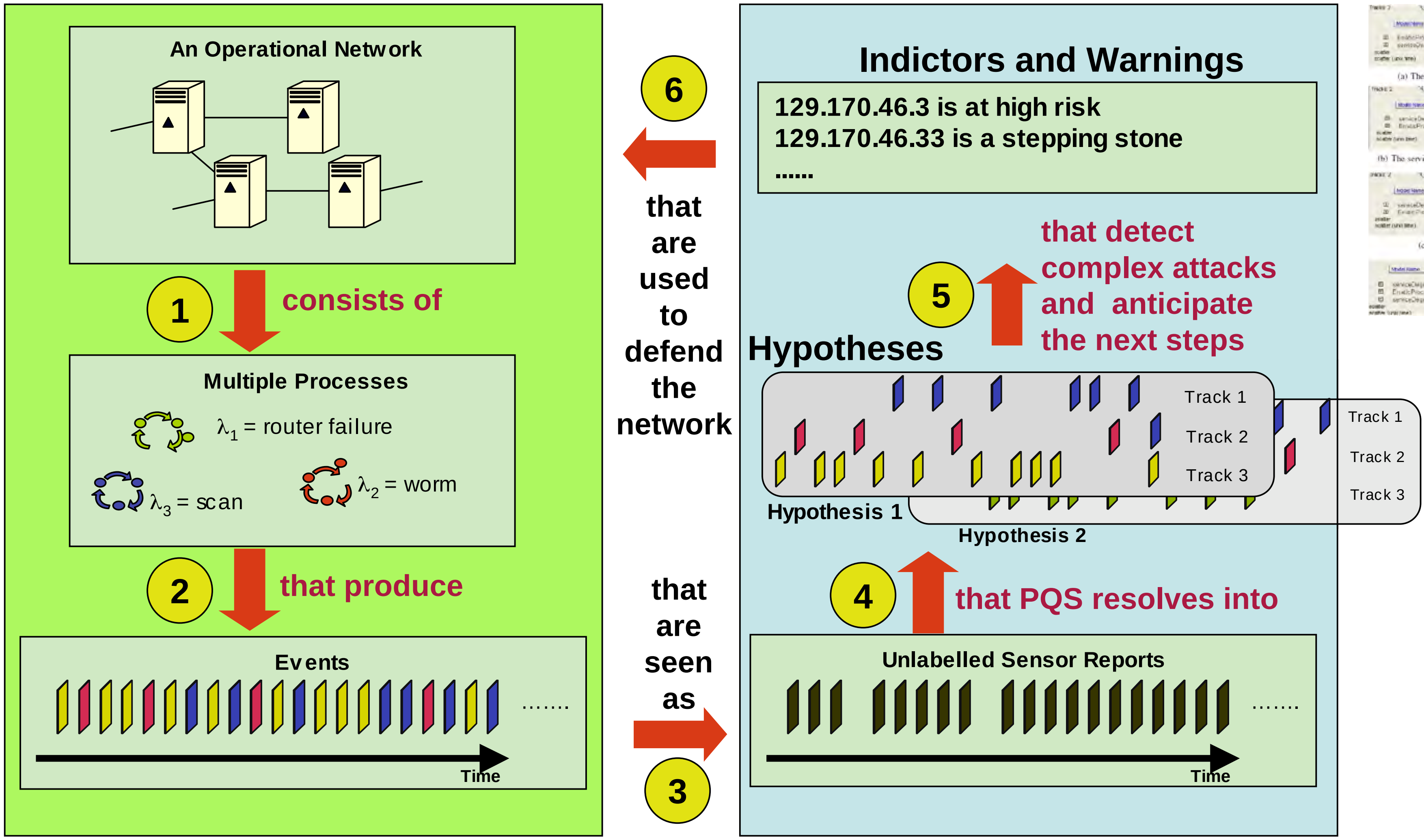
- **How it works:**
  - User provides a *process* description as query
  - PQS monitors a stream of sensor data
  - PQS matches sensor data with registered queries
  - A match indicates that the process model may explain that sensor data, hence that process may be the cause of those sensor readings.

Applications

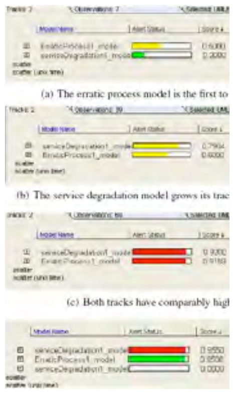
- ➡ **Tactical C4ISR** - Is there a large ground **vehicle** convoy moving towards our position?
- ➡ **Cyber-security** - Is there an unusual pattern of **network and system calls** on a server?
- ➡ **Autonomic computing** - Is my **software** operating normally?
- ➡ **Plume detection** – where is the source of a hazardous **chemical plume**?
- ➡ **FishNet** – how do **fish** move**?**
- **Insider Threat Detection** - Is there a pattern of unusual **document accesses** within the enterprise document control system?
- **Homeland Security -** Is there a pattern of unusual **transactions**?
- **Business Process Engineering** - Is the **workflow system** working normally?
- **Stock Market**
- **…**

All are "<u>adversarial</u>" processes, not cooperative so the observations are not necessarily labeled for easy identification and association with a process!

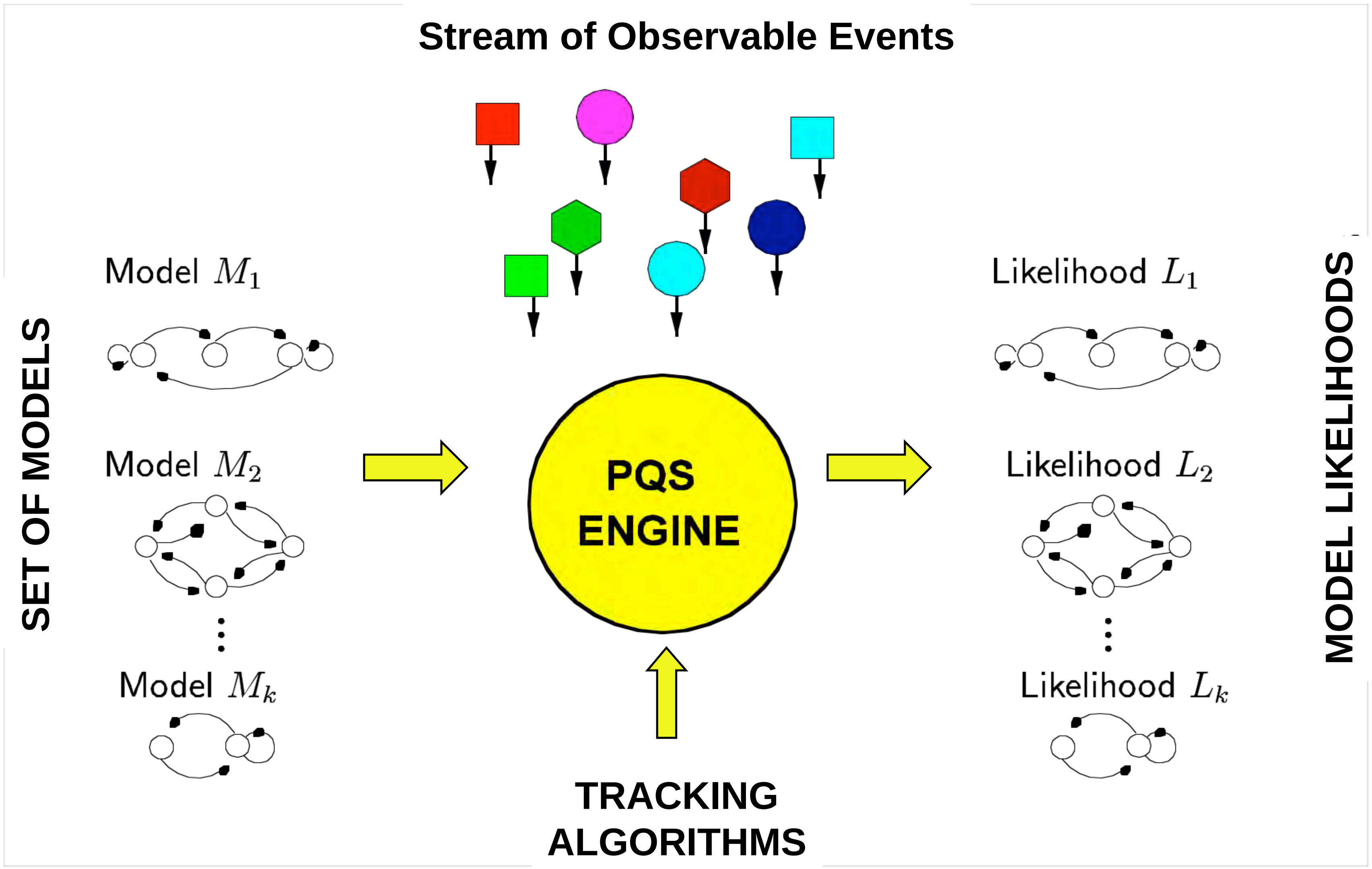
Example



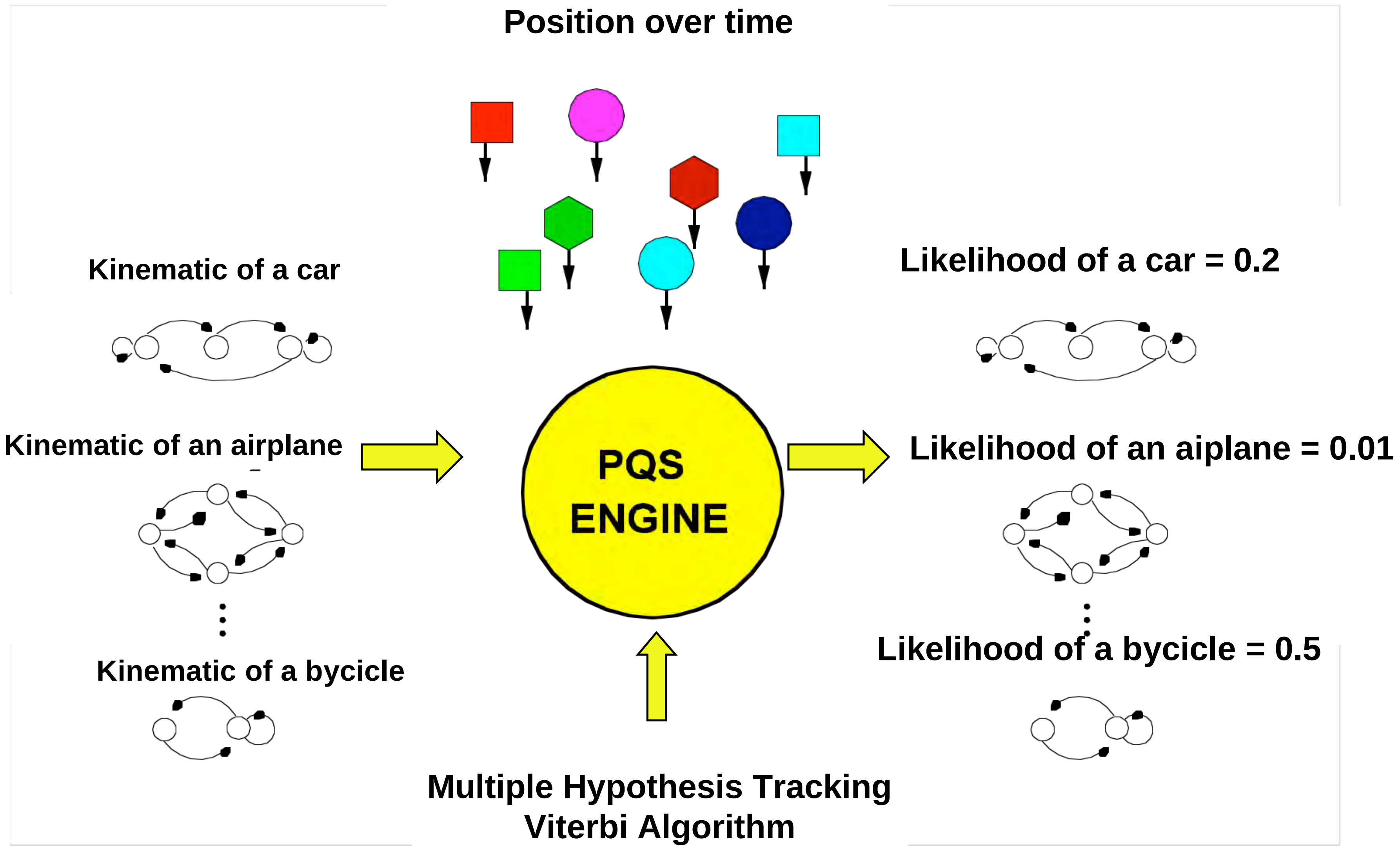
An Operational Network
1
consists of
Multiple Processes
λ_1 = router failure
λ_3 = scan
λ_2 = worm
2
that produce
Events
....
Time
that are seen as
3
Unlabelled Sensor Reports
....
Time
4
that PQS resolves into
Hypotheses
Track 1
Track 2
Track 3
Hypothesis 1
Hypothesis 2
5
that detect complex attacks and anticipate the next steps
Indictors and Warnings
129.170.46.3 is at high risk
129.170.46.33 is a stepping stone
......
6
that are used to defend the network



PQS


Stream of Observable Events
SET OF MODELS
Model M_1
Model M_2
Model M_k
PQS ENGINE
TRACKING ALGORITHMS
Likelihood L_1
Likelihood L_2
Likelihood L_k
MODEL LIKELIHOODS

PQS


Position over time
Kinematic of a car
Kinematic of an airplane
Kinematic of a bycicle
PQS ENGINE
Likelihood of a car = 0.2
Likelihood of an aiplane = 0.01
Likelihood of a bycicle = 0.5
Multiple Hypothesis Tracking
Viterbi Algorithm

OUTLINE

- Covert Channels
- Process Query Systems
- ➡ Detection of covert channels using a PQS

Observations

Time T

source ip:	**129.170.248.33**
dest ip:	**208.253.154.210**
source port:	**44806**
dest port:	**23164**

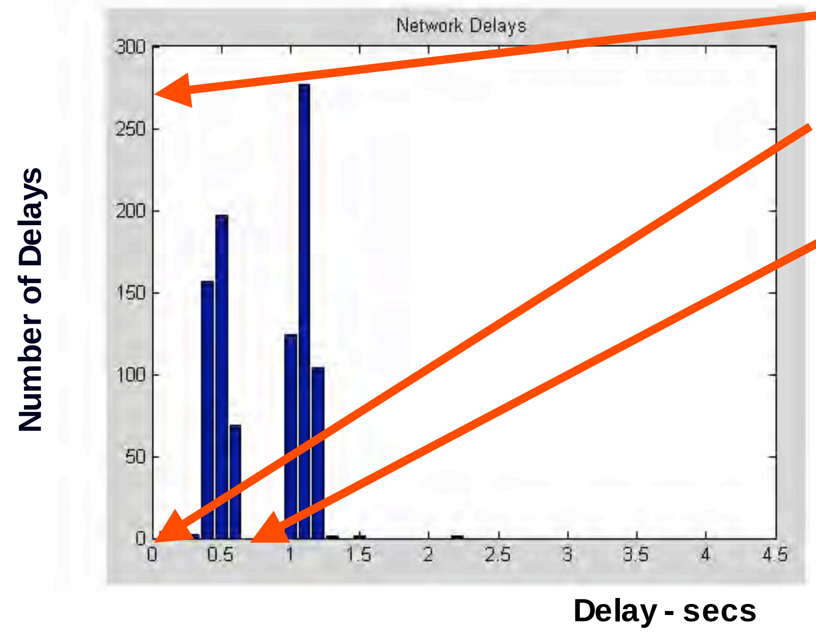
$$C_{mean} / C_{\max}$$

Time T+1

source ip:	**129.170.248.33**
dest ip:	**208.253.154.210**
source port:	**44806**
dest port:	**23164**

$$C_{mean} / C_{\max}$$

Covert Channels models

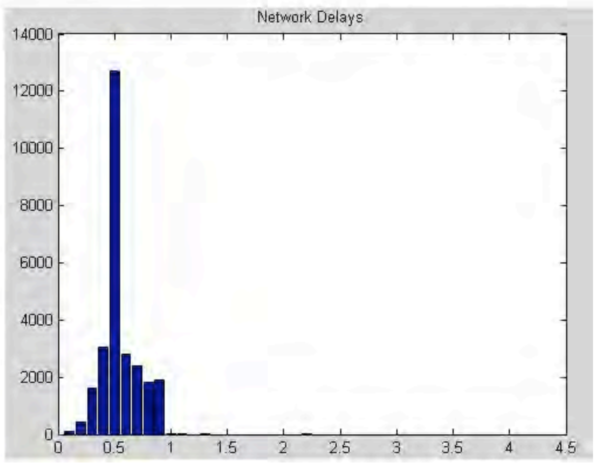


$C_{\max}$ = max number of packets with the same delay (= 280)

$C(\hat{\imath}\ )$ = number of packets with interpacket delay (= 0)

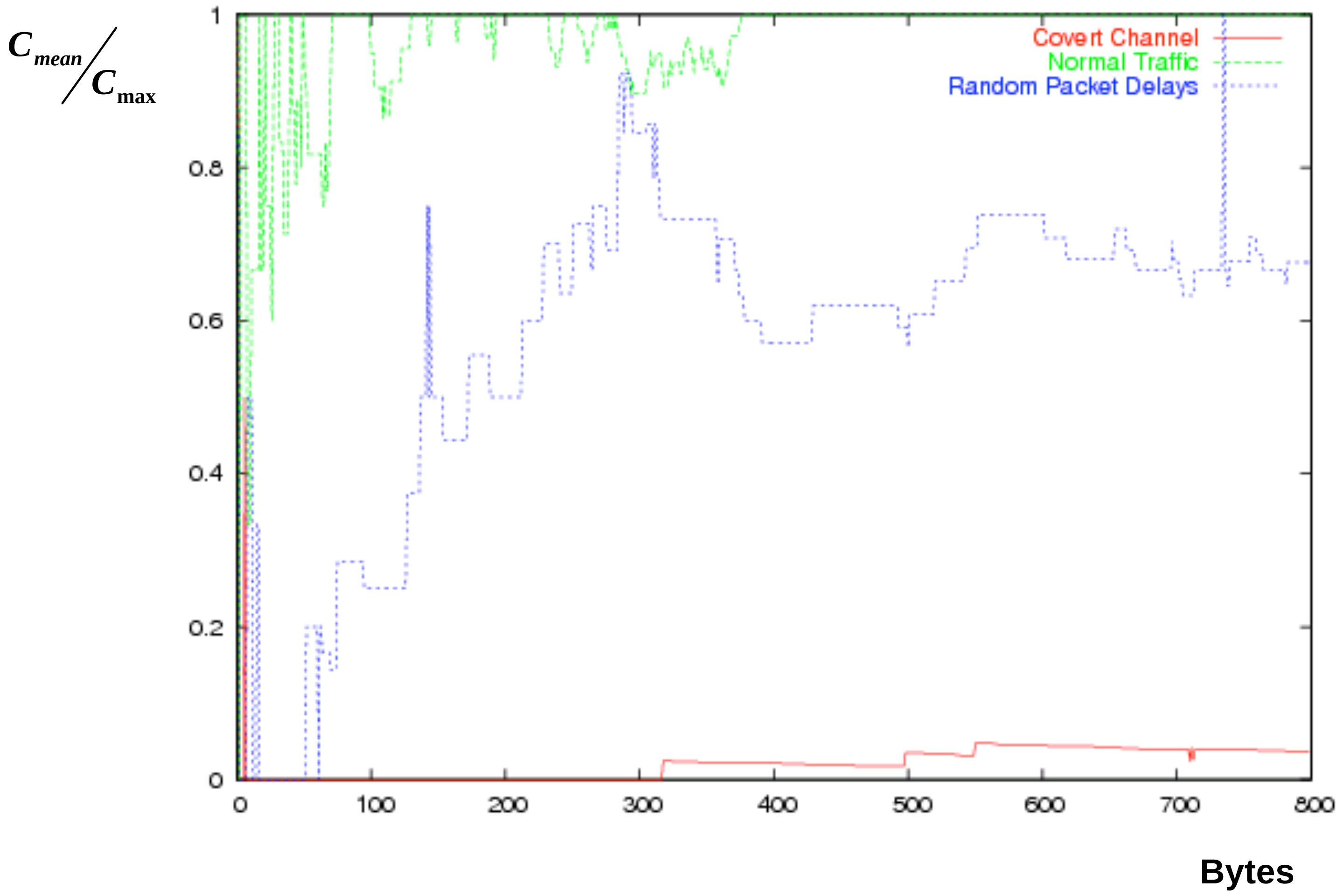
μ = sample mean of interpacket transmission times (= 0.7)

$\frac{C(\mu)}{C_{\max}} \approx 1$ ➡ **Not Covert Channel**



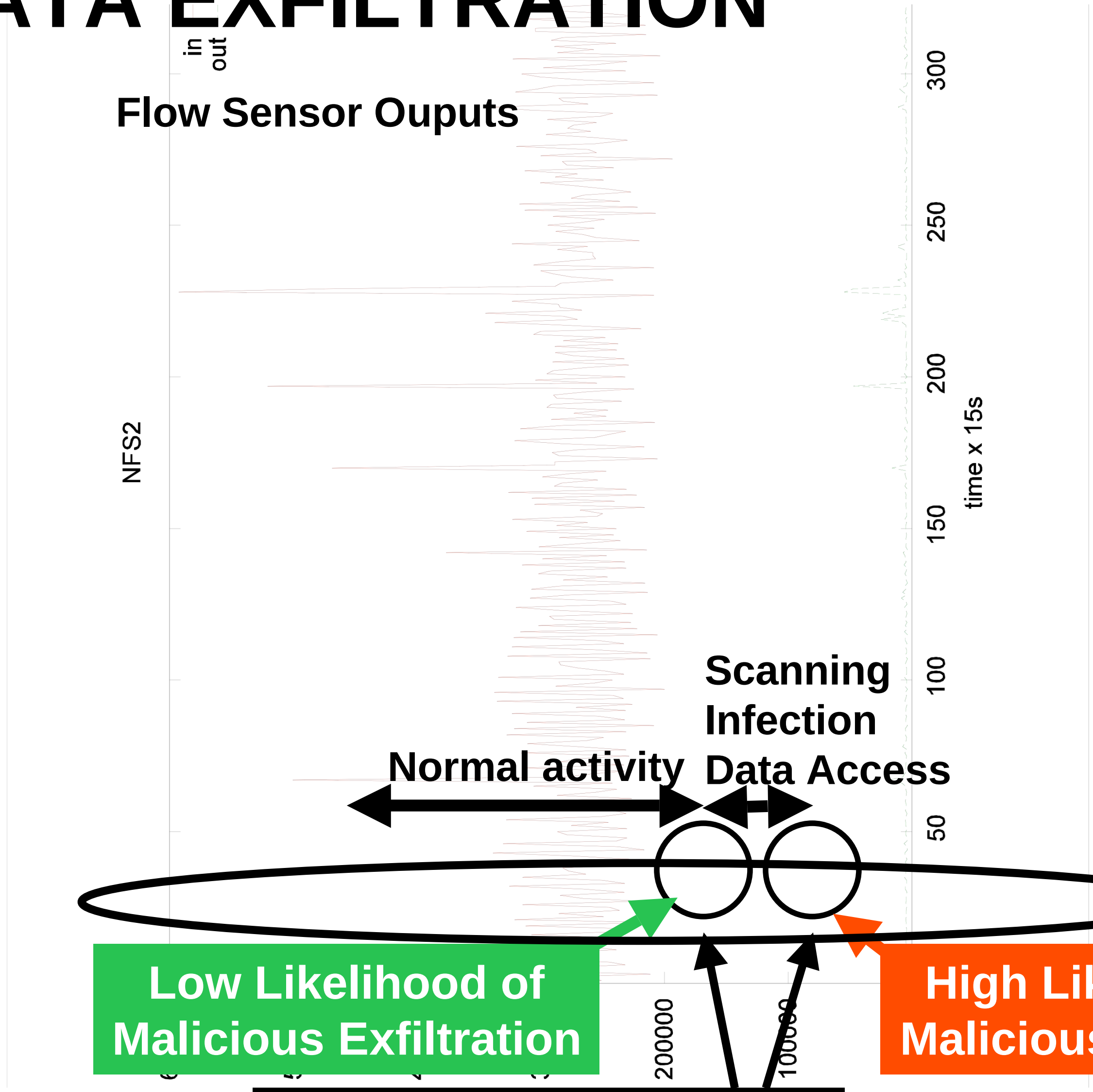
$\frac{C(\mu)}{C_{\max}} << 1$ ➡ **Covert Channel**

RESULTS



C_{mean}/C_{max}
1
0.8
0.6
0.4
0.2
0
0
100
200
300
400
500
600
700
800
Covert Channel
Normal Traffic
Random Packet Delays
Bytes

DATA EXFILTRATION



Low Likelihood of Malicious Exfiltration

Increased outbound data

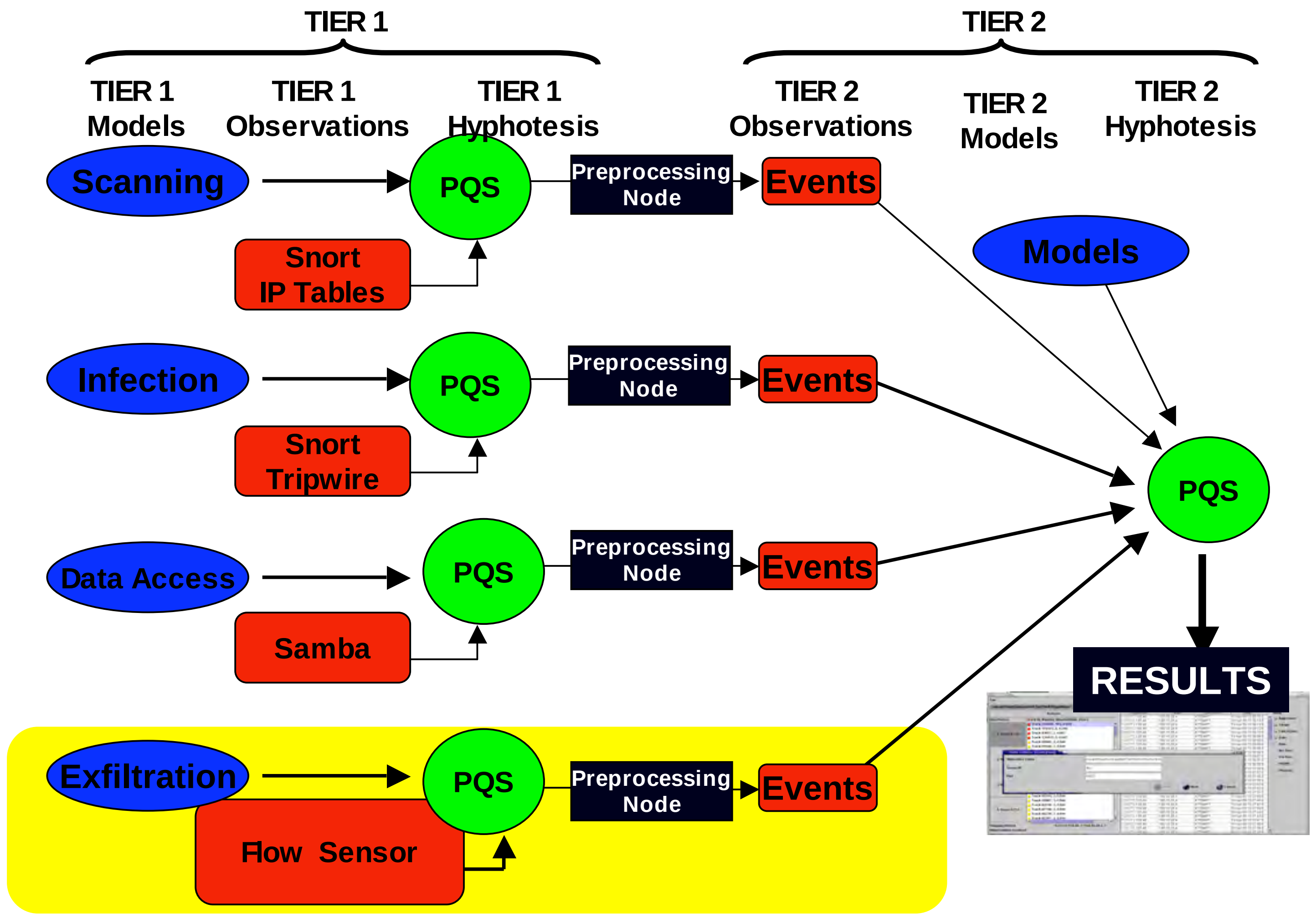
High Likelihood of Malicious Exfiltration

Exfiltration modes:

- SSH
- HTTP
- FTP
- Email
- Covert Channel
- Phishing
- Spyware
- Pharming
- Writing to media
  - paper
  - drives
- etc

Also monitor inter-packet delays for covert channels

Hierarchical PQS Architecture



For more information :

www.pqsnet.net
www.ists.dartmouth.edu

annarita.giani@dartmouth.edu

vincent.berk@dartmouth.edu

george.cybenko@dartmouth.edu

Thanks.

Correlations between quiescent ports in network flows

Joshua McNutt & Markus De Shon
{jmcnutt,mdeshon}@cert.org
September 21, 2005

What is a quiescent port?

A TCP or UDP port not in regular use

- No assigned service
- Obsolete service
- Ephemeral port with no active service

Port summary data

- Flows too detailed for some analysis
- Full flow data huge, slow interactive analysis
- Which flows are of interest?
- Therefore: Hourly summaries populate a database
  - # Flows
  - # Packets
  - # Bytes
  - Per port (TCP/UDP)
  - Per ICMP Type and Code
  - Per IP Protocol
  - “Incoming” and “Outgoing”

Anomaly detection

There are many kinds of “anomaly detection”

Here we mean: statistical anomaly detection

Problem: Network data does not behave

- Self-similarity
- “Infinite” variance
- Not normal distributions

Problem: Data is noisy

- Vertical scanning
- Return traffic from web requests, outgoing email
- Other behavior masked

Correlation

•Our realization:

•Vertical scanning leads to correlations between server ports

•Web & email return traffic leads to correlations between ephemeral ports

•Other kinds of activity may concentrate on only one port

- Horizontal scanning
- Backdoor activity
- Worms

Robust correlation

Any anomaly detection method has a problem:

- What if the activity of interest occurs during the learning period?
- The model of “normal” is skewed

Solution: exclude the outliers

“Robust correlation”

- Exclude 5% most extreme outliers (Rousseew and Van Zomeren 1990)
- Calculate correlations based on remainder

Robust correlation matrix

Take time series for ports (e.g. 0-1023)

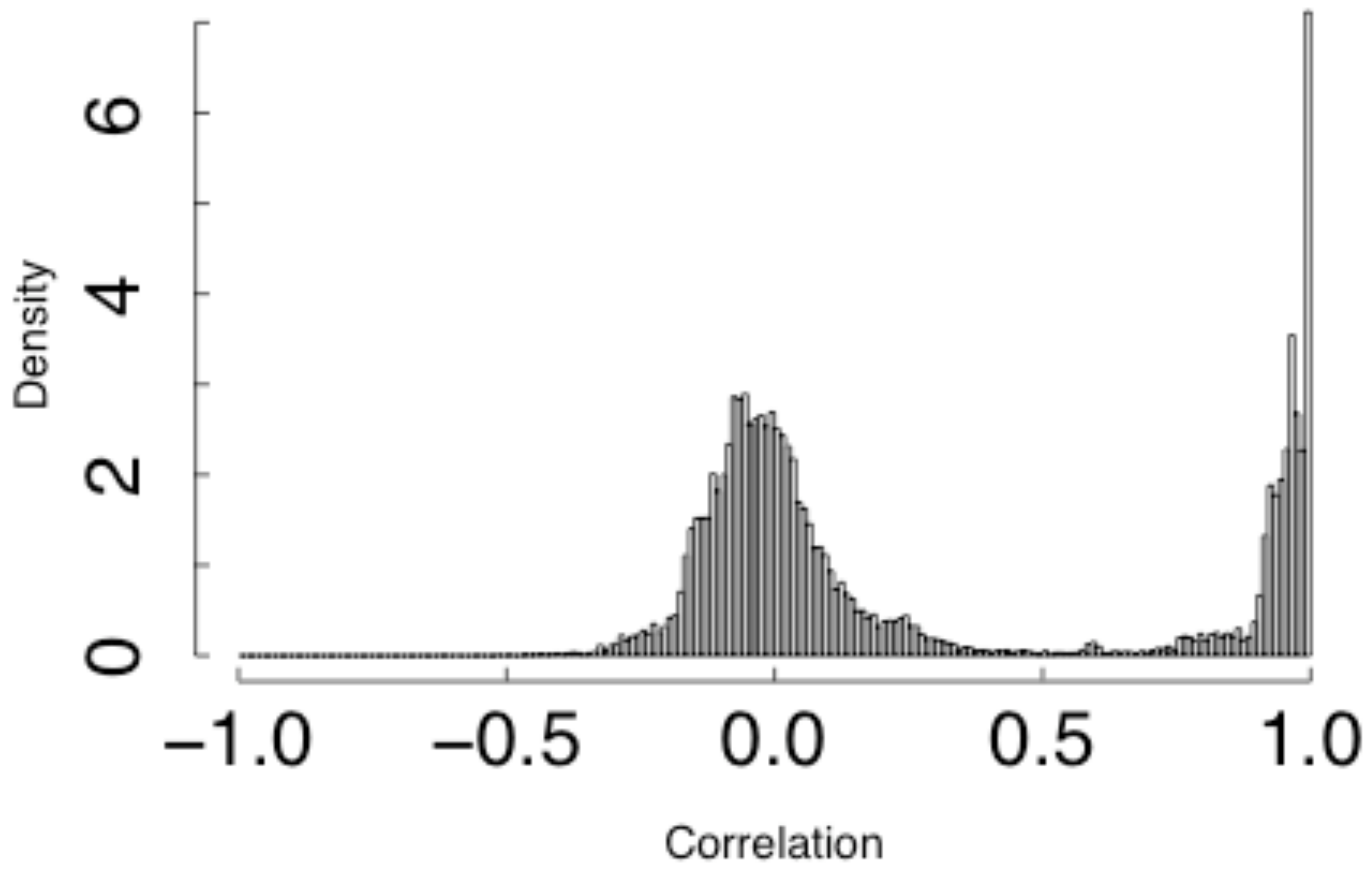
Calculate every robust correlation C(i,j)

C(i,j) is symmetric, and diagonal == 1

- C(i,i) == 1
- C(i,j) == C(j,i)

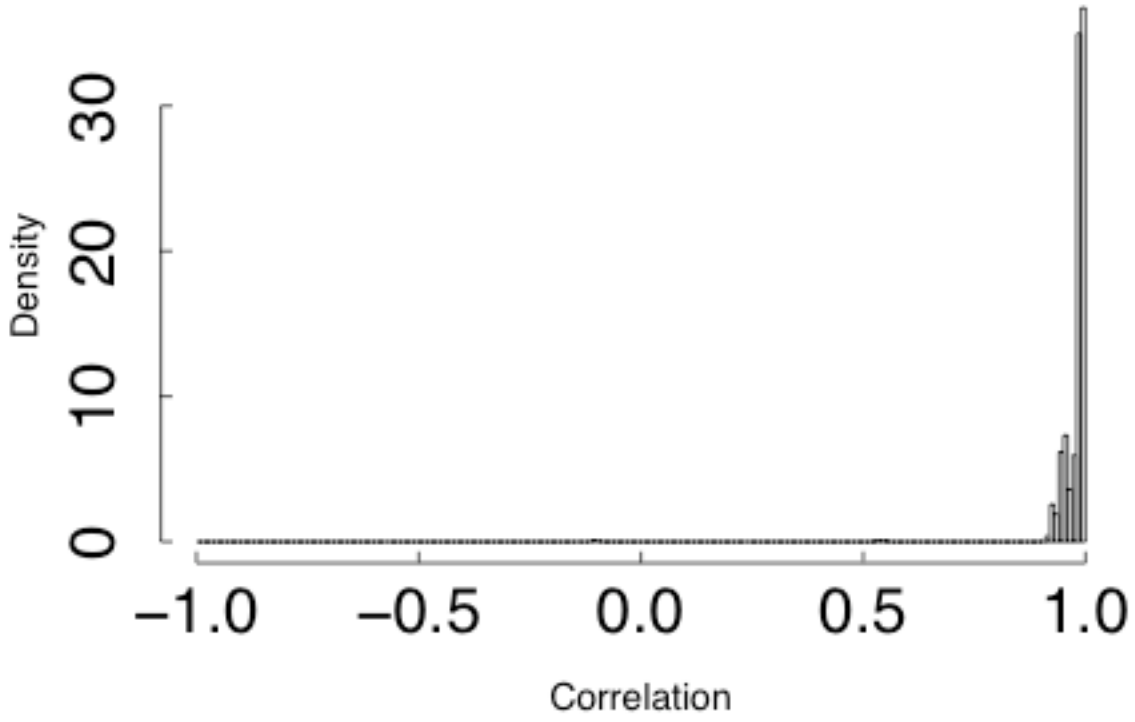
Robust correlation distribution

TCP ports 0-1023



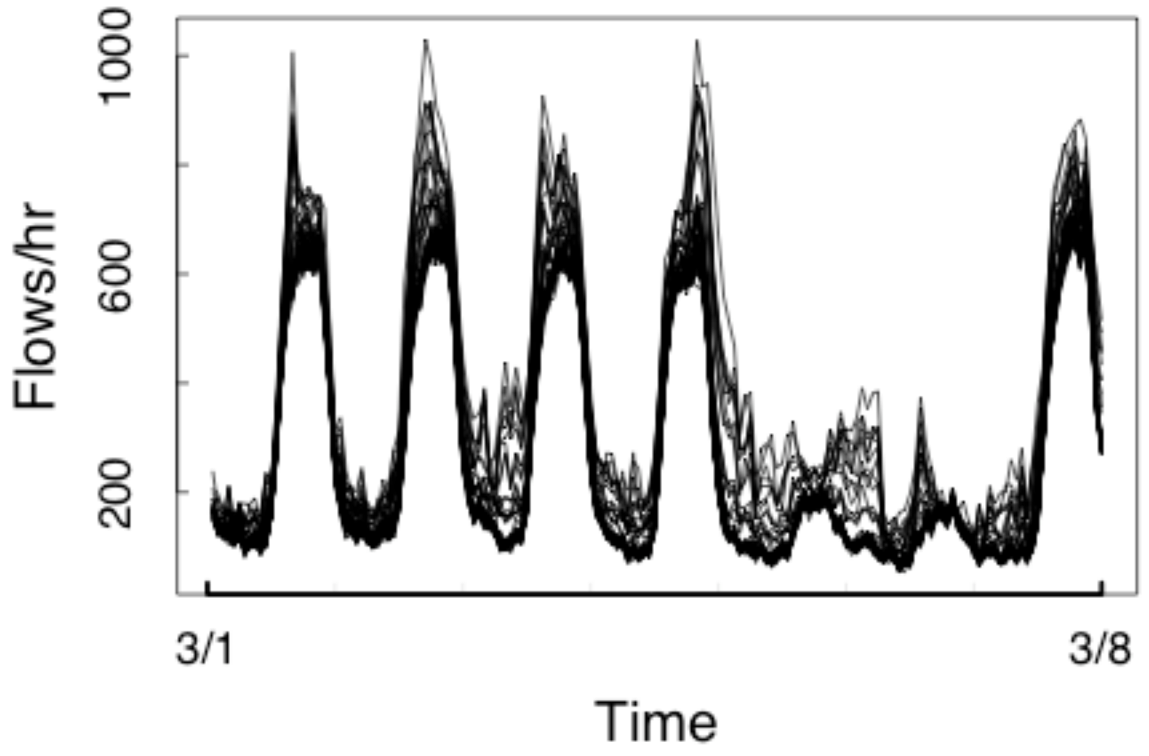
Ephemeral port correlations (cont'd)

Robust correlation distribution (TCP/50000-51024)



Ephemeral port correlations

50 high numbered ports



Correlation clusters

Many correlated ports (indicated by ::)

If A::B and B::C, then A::C

Can we identify clusters A::B::C::D::…

Yes!

- For 0-1023, cluster of 133 ports
  - Could be higher with better data (need to include filtered traffic)
- For 1024+, nearly all ports are correlated
  - Large number of independent web browsers lead to well-behaved seasonality

Server ports

Ports 0-1023

Generally servers

Many unassigned/unused ports

Lots of filtering

Some obsolete services, possible source of threats

Ephemeral ports

Ports 1024-65535

A few servers

- Databases (Oracle 1521, MS SQL 1433/1434)
- Proxies (1080/8080)
- RPC services

Peer-to-peer

Backdoors (31337, etc)

Ephemeral ports for client services

- Request/response results in *two* flows

The Method

Identify correlation cluster

Monitor all clustered ports, detect deviations

- Find median flow count for cluster, subtract from each port
- Significant number of flows above median → alert

Investigate deviations further

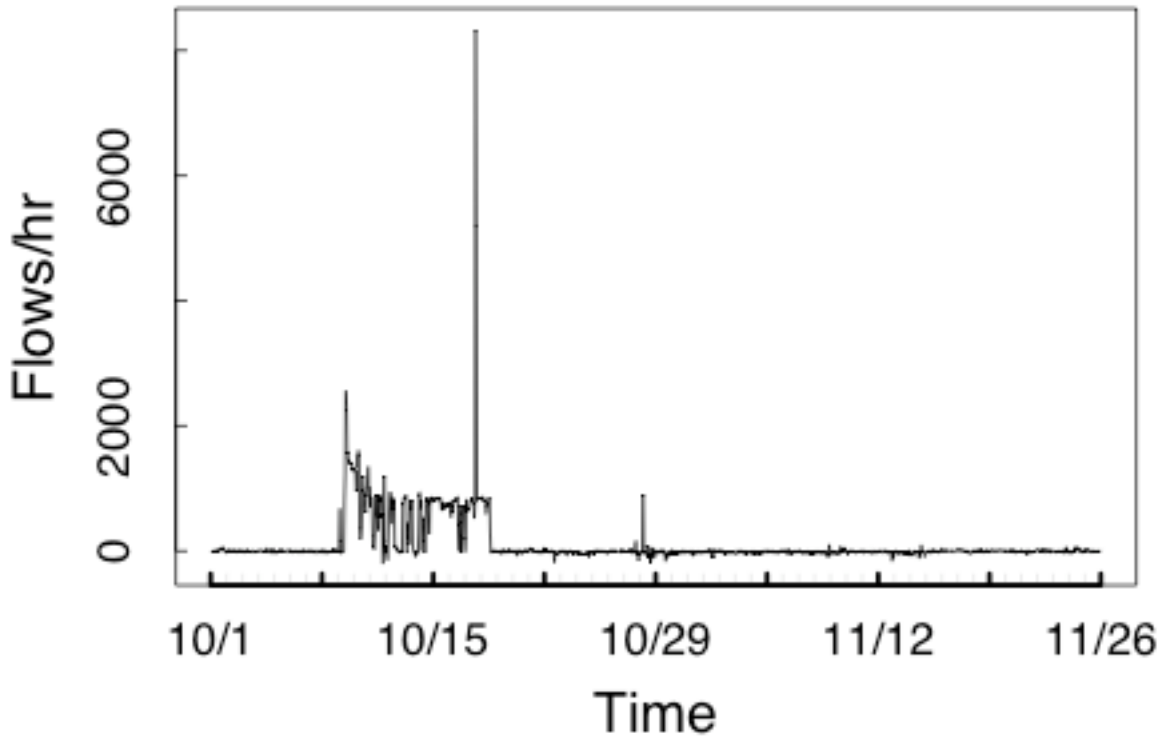
- Increased flows + increased hosts, intermittent → widespread horizontal scanning
- Increased flows + increased hosts, persistent → possible worm
- Increased flows, no increased hosts → localized activity, possibly still a threat

Case Study: 42/TCP

- Microsoft Windows Internet Name Service (WINS)
- Phasing out (replaced by Active Directory, DNS)
- Still present in Win2k3 Server
- Vulnerability announced Nov 25, 2004
- Scanning publicly announced Dec 12
- Could we have detected scanning earlier?

42/TCP: Deviations from correlation

Before vulnerability announcement



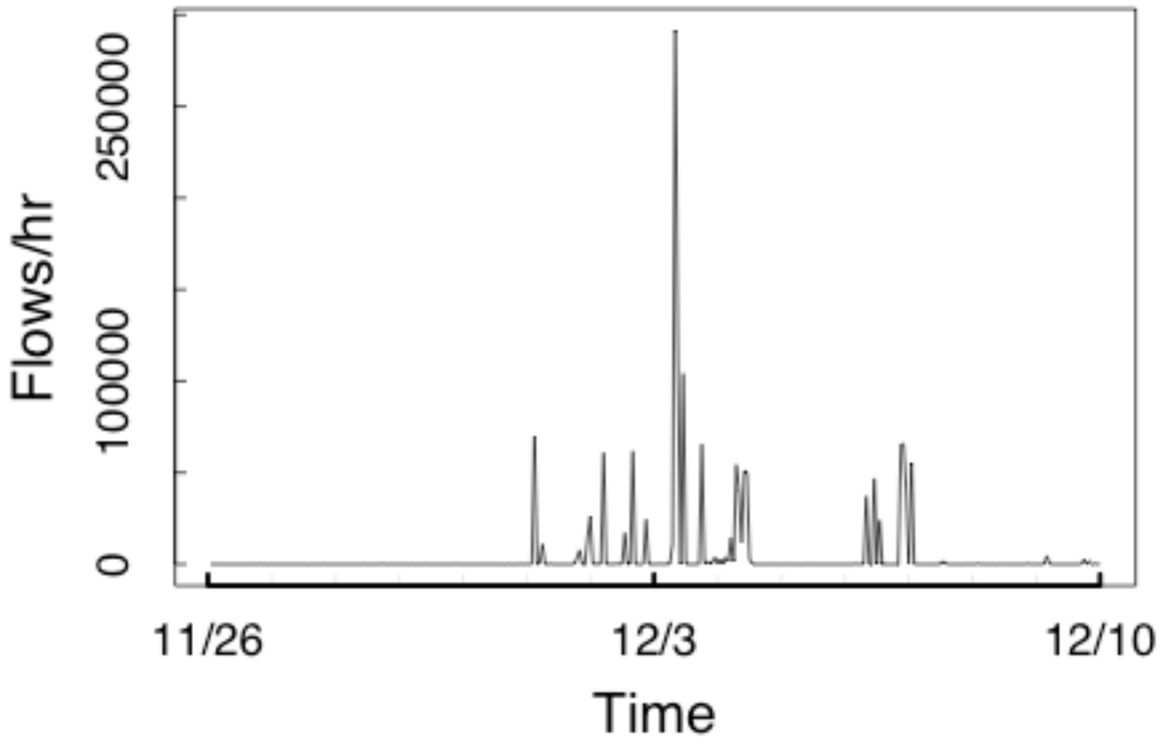
Software Engineering Institute

42/TCP: Deviations before vulnerability announcement

- Some deviations observed
- Always involved a small number of hosts (1 or 2)
- < 10,000 additional flows/hour
- No global activity indicated

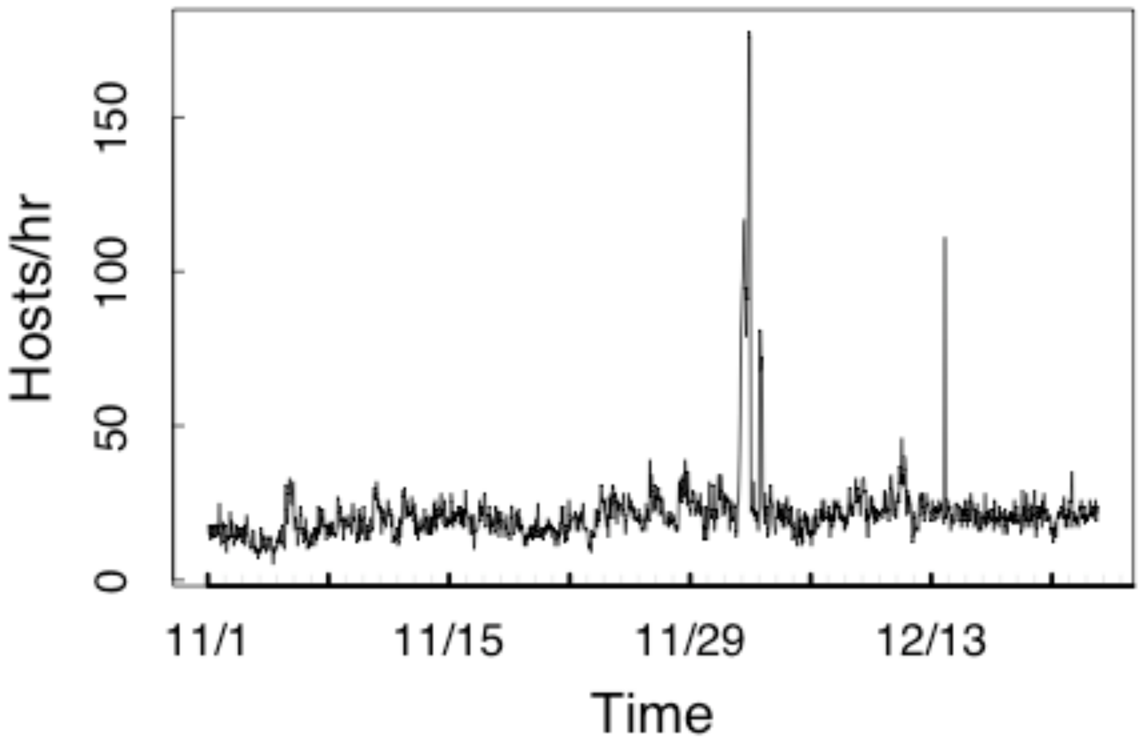
42/TCP: Deviations from correlation

After vulnerability announcement, # flows/hr



42/TCP: Deviations from correlation

After vulnerability announcement, # hosts/hr

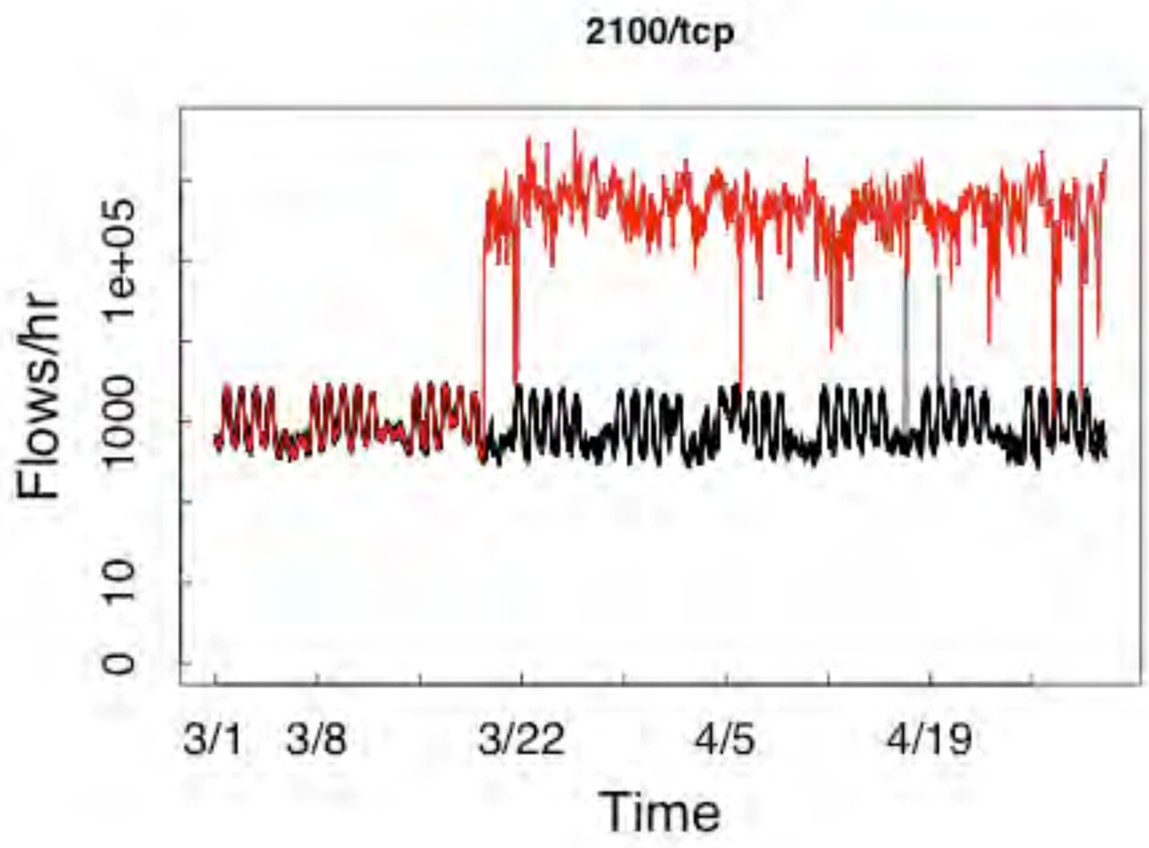


42/TCP: Deviations from correlation

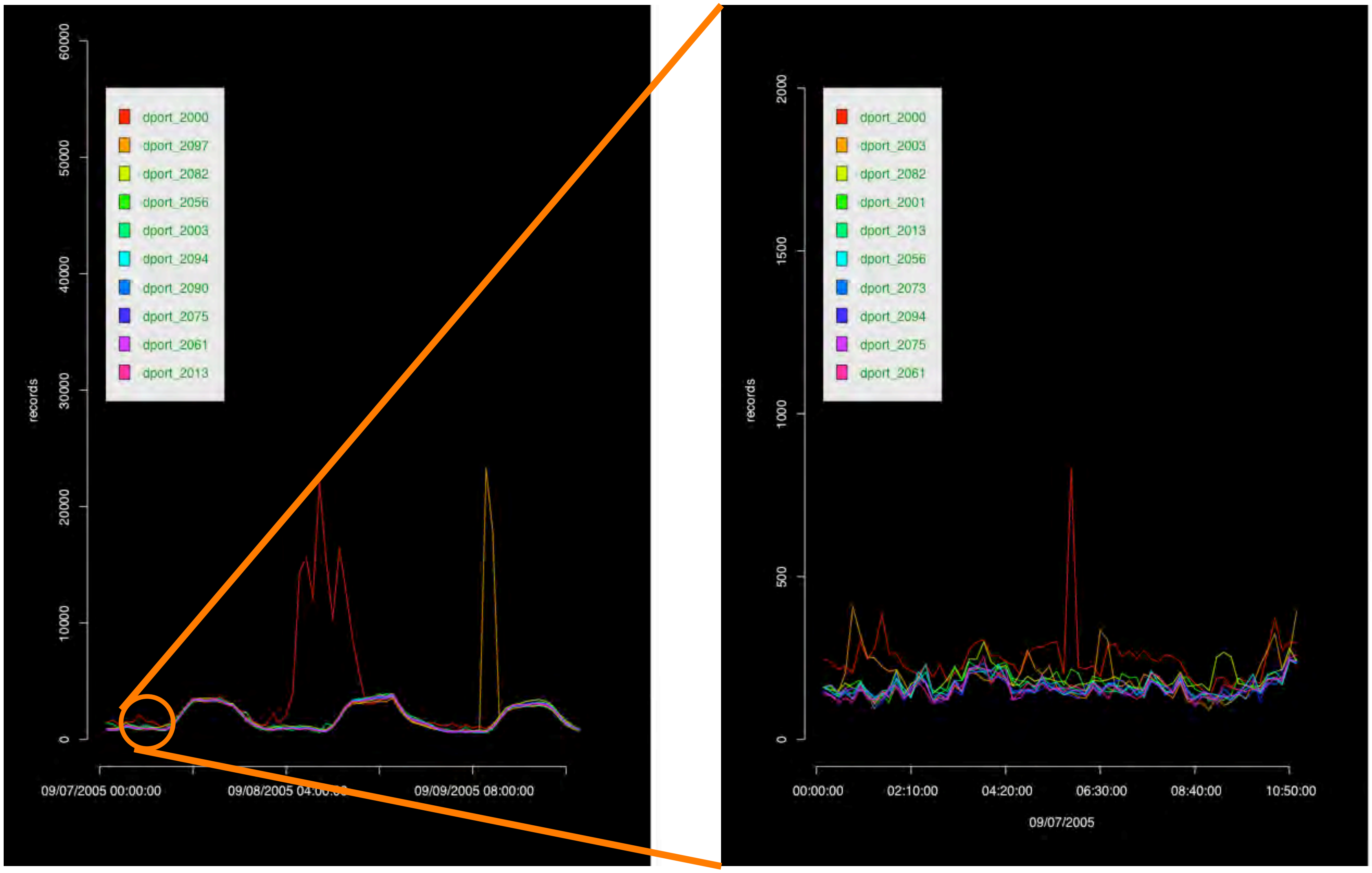
After the announcement on 11/25

- Large increase in flows 12/1 2am (>100,000 additional flows/hr)
- Surge in #hosts/hr by 12/1 midnight
- Could have announced:
  - Scanning of port 42/TCP observed
  - Announce by morning of 12/2
  - Ahead of other announcement by 10 days

Port 2100/TCP



Interactive analysis



Future Directions

Median in sliding window of ports?

- Uncover attacks against ranges of ports

Unique number of sources, destinations

- ipsets?

Work on non-quiescent ports

- Some experiences with ephemeral ports (return traffic)
- Models will differ for different services
  - user-driven (e.g. web)
  - automated (e.g. ntp)

Flows vs. bytes vs. packets

- Peer-to-peer
- Information exfiltration

Automatic identification of backscatter (to be ignored?)

Conclusions

Many ports highly correlated

- Vertical scanning (esp. server ports)
- Client activity responses (ephemeral ports)

Removing correlated activity exposes other activity

- DDoS backscatter
- Port-specific scanning
- Port-specific exploit attempts
- Worms

42/TCP real world example

- Clear signal
- Public announcement 10 days earlier

Automated method for focusing attention on specific ports

CERT/NetSA

CERT/NetSA

Software Engineering Institute

Carnegie Mellon University

4500 Fifth Avenue

Pittsburgh PA 15213

USA

Web: http://www.cert.org/netsa

Flow-based Analysis

A *flow* is a one-way network traffic instance

- Source ip and port → destination IP and port
- Corresponds to 1 side of a TCP session
- Aggregates UDP pseudo-sessions
- Times out

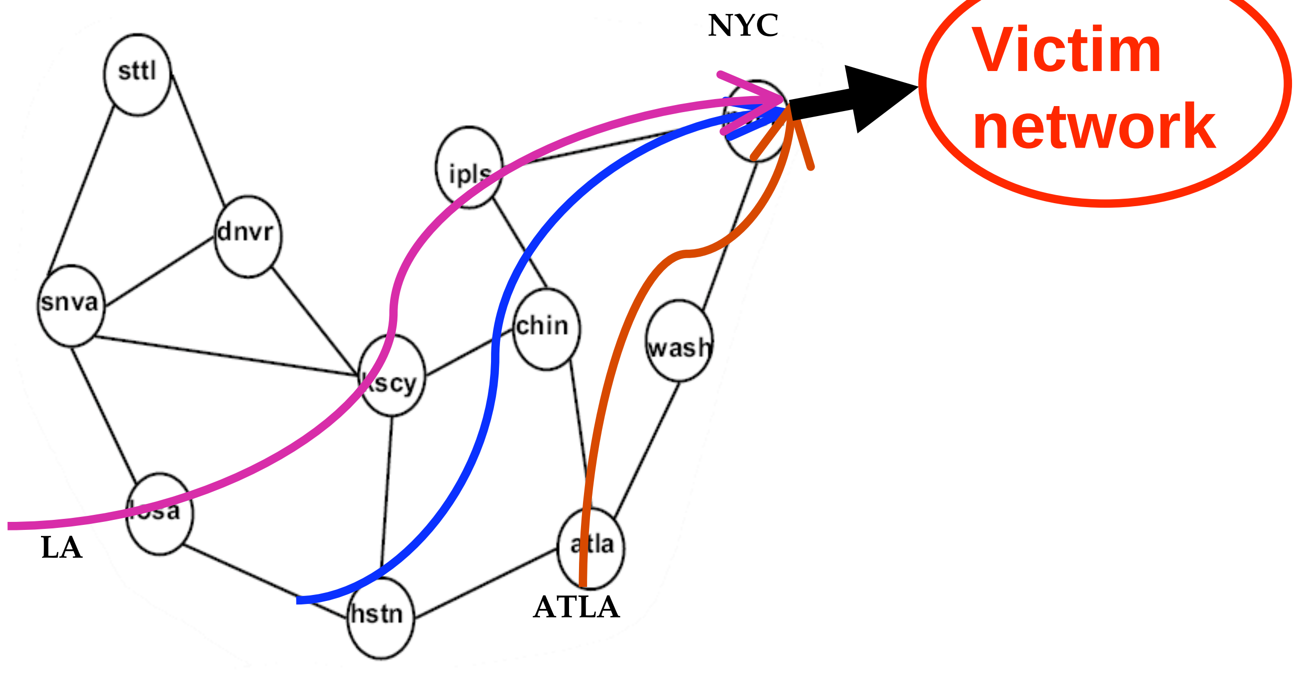
Example implementation: Cisco NetFlow

Detecting Distributed Attacks Using **Network-Wide Flow Data**

Anukool Lakhina
with Mark Crovella and Christophe Diot

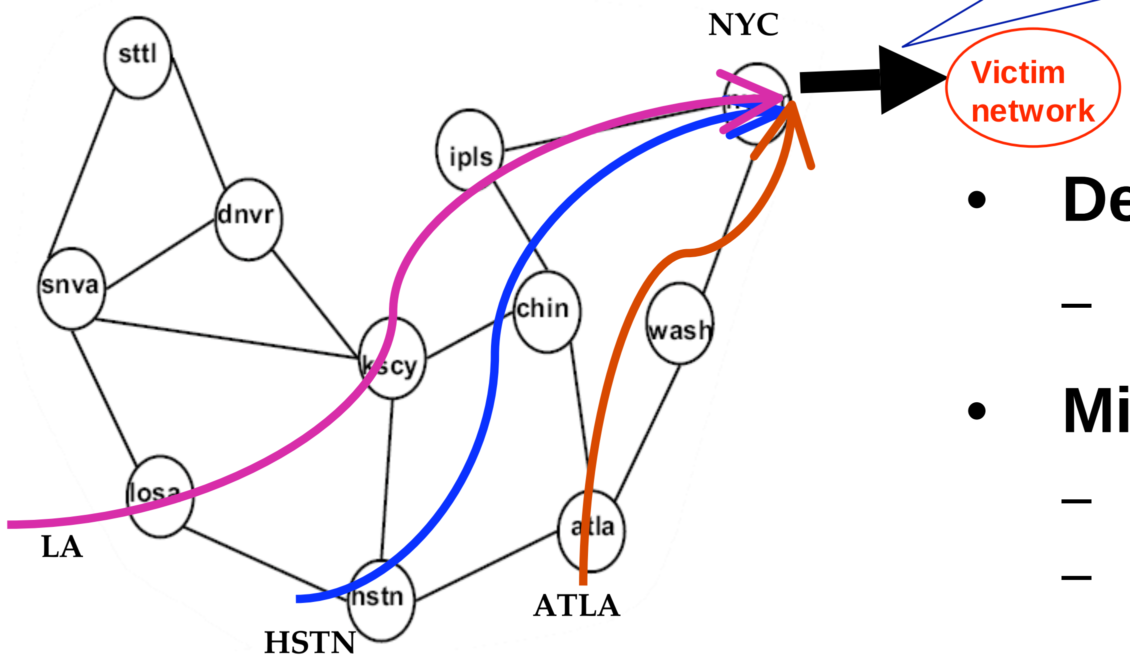
FloCon, September 21, 2005

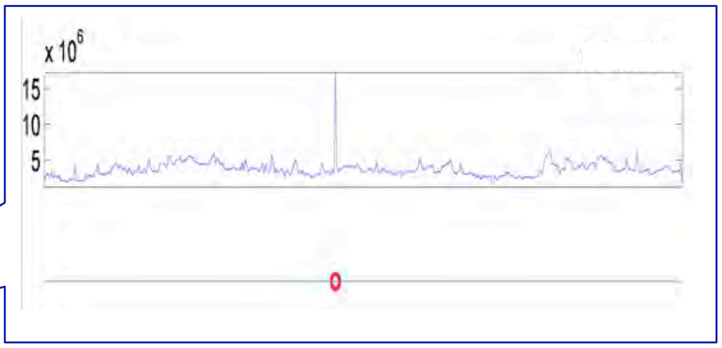
The Problem of Distributed Attacks



- Continue to become more prevalent [CERT'04]
- Financial incentives for attackers, *e.g.*, extortion
- Increasing in sophistication: worm-compromised hosts and bot-nets are massively distributed

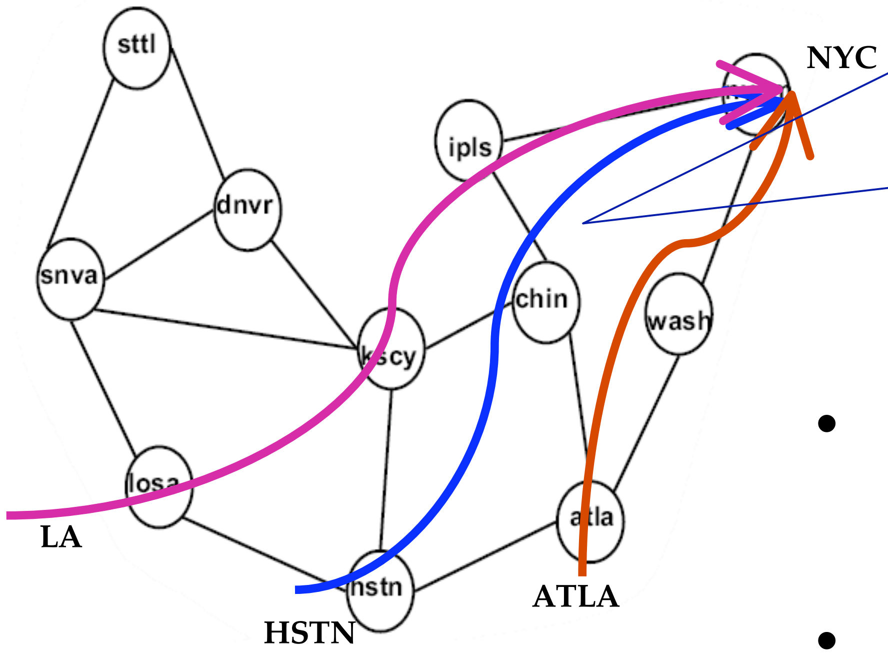
Detection at the Edge

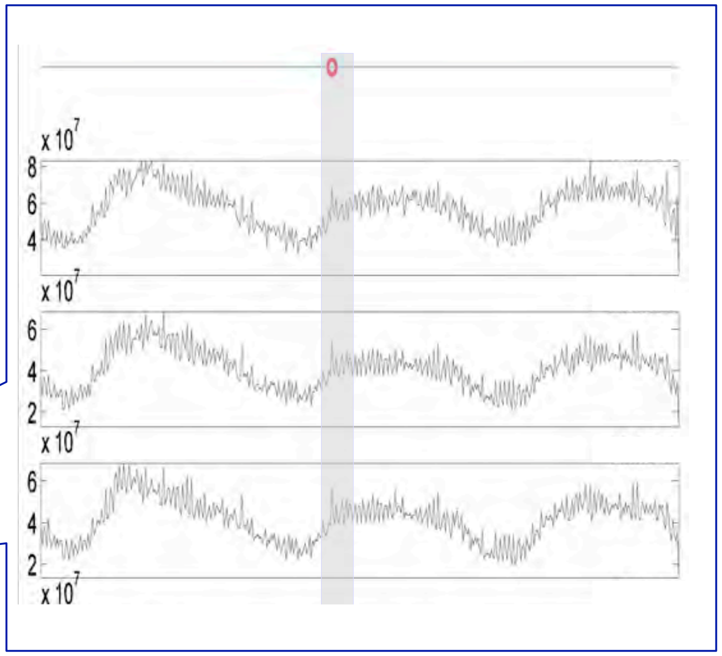




- **Detection easy**
  - Anomaly stands out visibly
- **Mitigation hard**
  - Exhausted bandwidth
  - Need upstream provider's cooperation
  - Spoofed sources

Detection at the Core

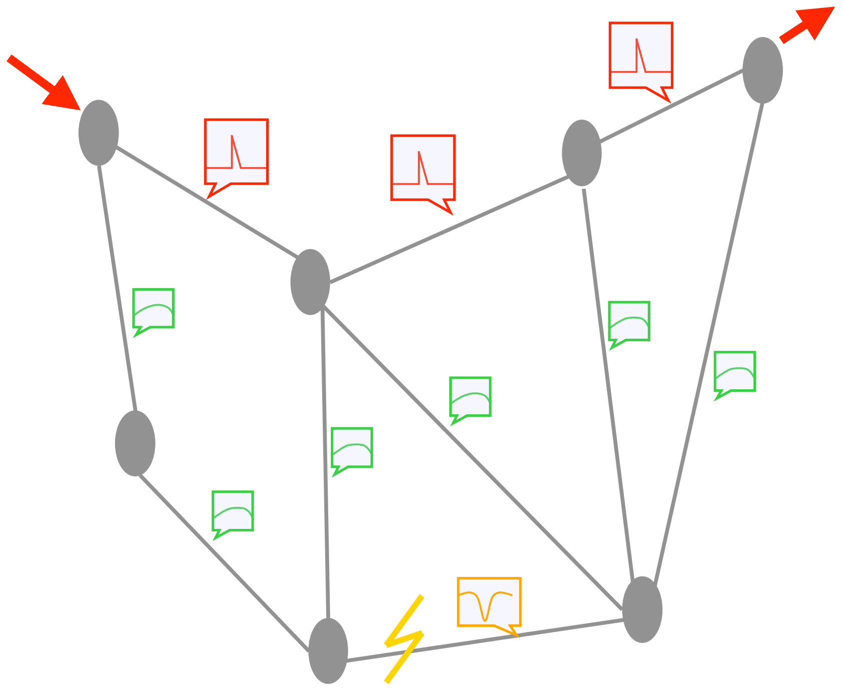




- **Mitigation Possible**
  - Identify ingress, deploy filters
- **Detection hard**
  - Attack does not stand out
  - Present on multiple flows

A Need for Network-Wide Diagnosis

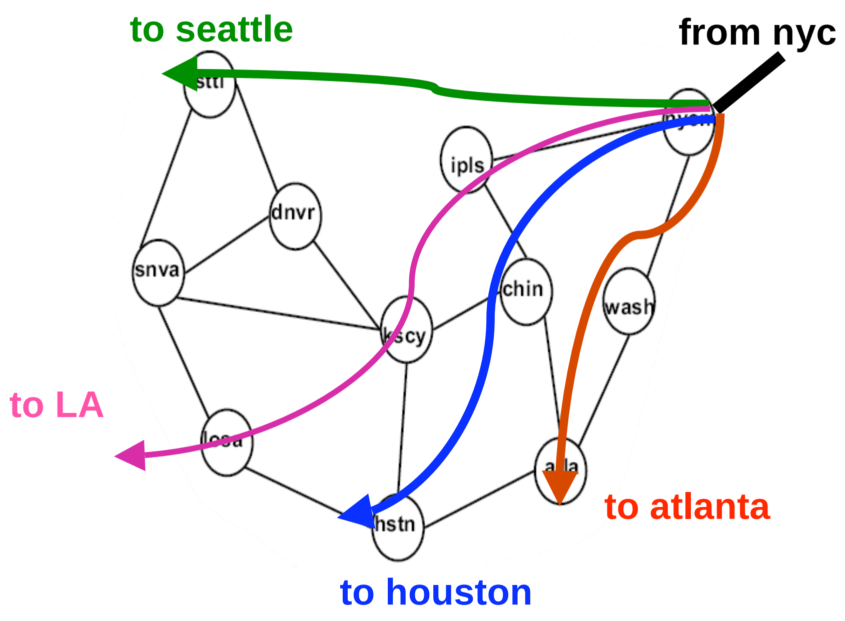
- Effective diagnosis of attacks requires a **whole-network approach**
  - *Simultaneously* inspecting traffic on all links
- Useful in other contexts also:
  - Enterprise networks
  - Worm propagation, insider misuse, operational problems



Talk Outline

- Methods
  - Measuring Network-Wide Traffic
  - Detecting Network-Wide Anomalies
  - Beyond Volume Detection: *Traffic Features*
  - Automatic Classification of Anomalies
- Applications
  - General detection: scans, worms, flash events, …
  - Detecting Distributed Attacks
- Summary

Origin-Destination Traffic Flows



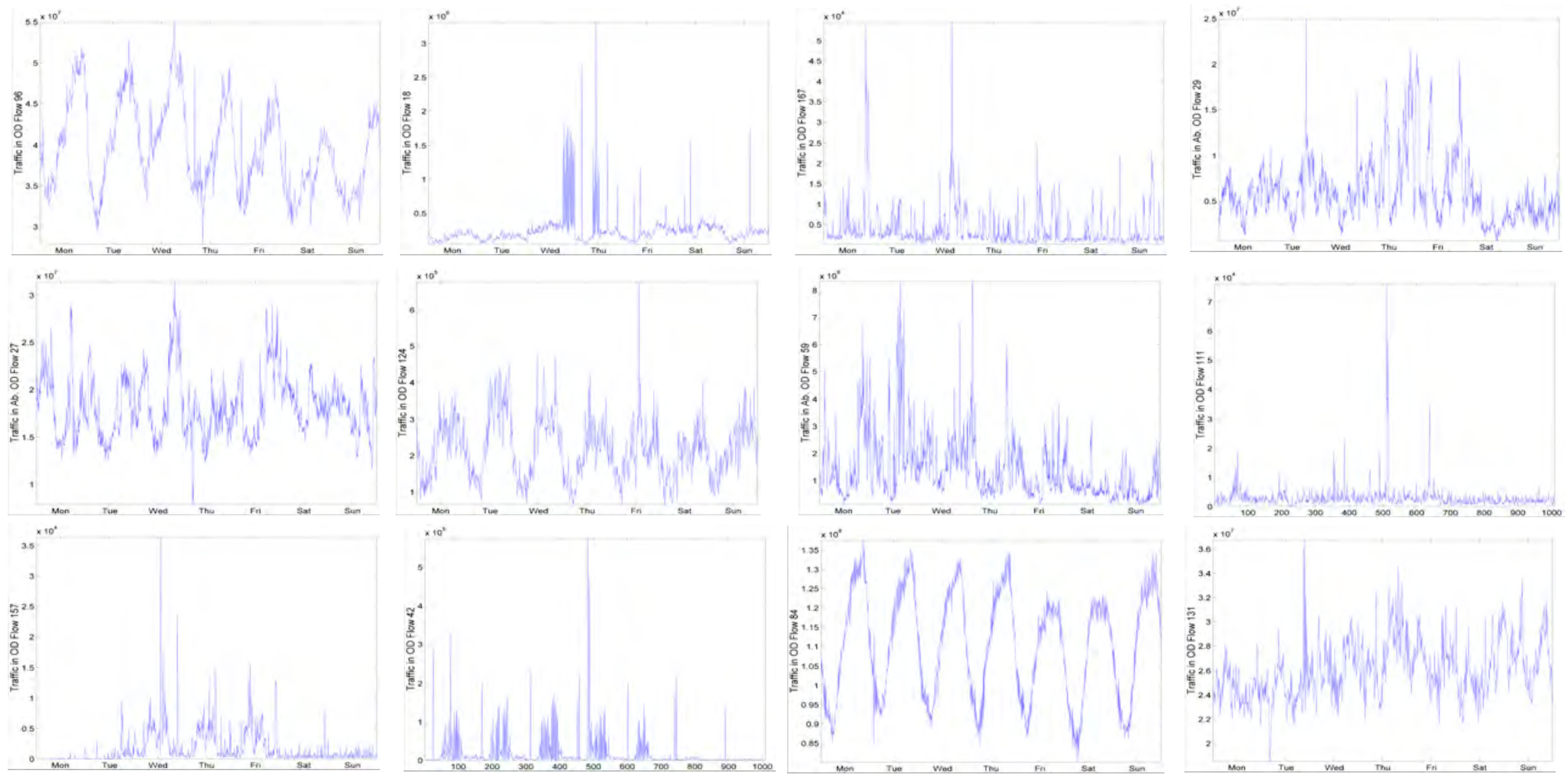
- Traffic entering the network at the *origin* and leaving the network at the *destination* (*i.e.*, the traffic matrix)
- Use routing (IGP, BGP) data to aggregate NetFlow traffic into OD flows
- *Massive* reduction in data collection

Data Collected

Collect sampled NetFlow data from all routers of:

1. **Abilene Internet 2 backbone research network**
   - 11 PoPs, 121 OD flows, anonymized,
     1 out of 100 sampling rate, 5 minute bins
2. **Géant Europe backbone research network**
   - 22 PoPs, 484 OD flows, not anonymized,
     1 out of 1000 sampling rate, 10 minute bins
3. **Sprint European backbone commercial network**
   - 13 PoPs, 169 OD flows, not anonymized,
     aggregated, 1 out of 250 sampling rate, 10
     minute bins

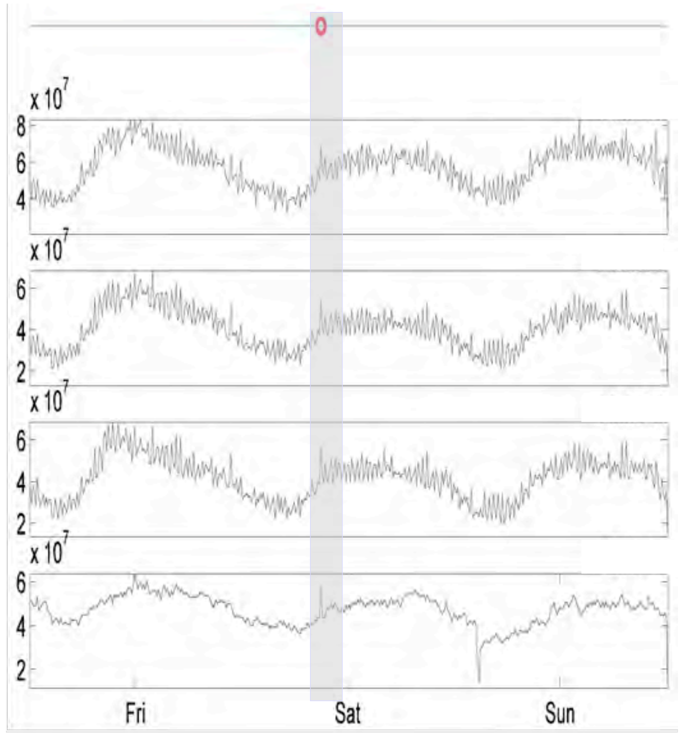
But, This is Difficult!



How do we extract **anomalies** and **normal behavior** from noisy, **high-dimensional** data in a systematic manner?

Turning High Dimensionality into a Strength

- Traditional traffic anomaly diagnosis builds normality in *time*
  - Methods exploit temporal correlation
- Whole-network view is an attempt to examine normality in *space*
  - Make use of spatial correlation
- Useful for anomaly diagnosis:
  - Strong trends exhibited throughout network are likely to be *“normal”*
  - Anomalies break relationships between traffic measures



The Subspace Method [LCD:SIGCOMM ‘04]

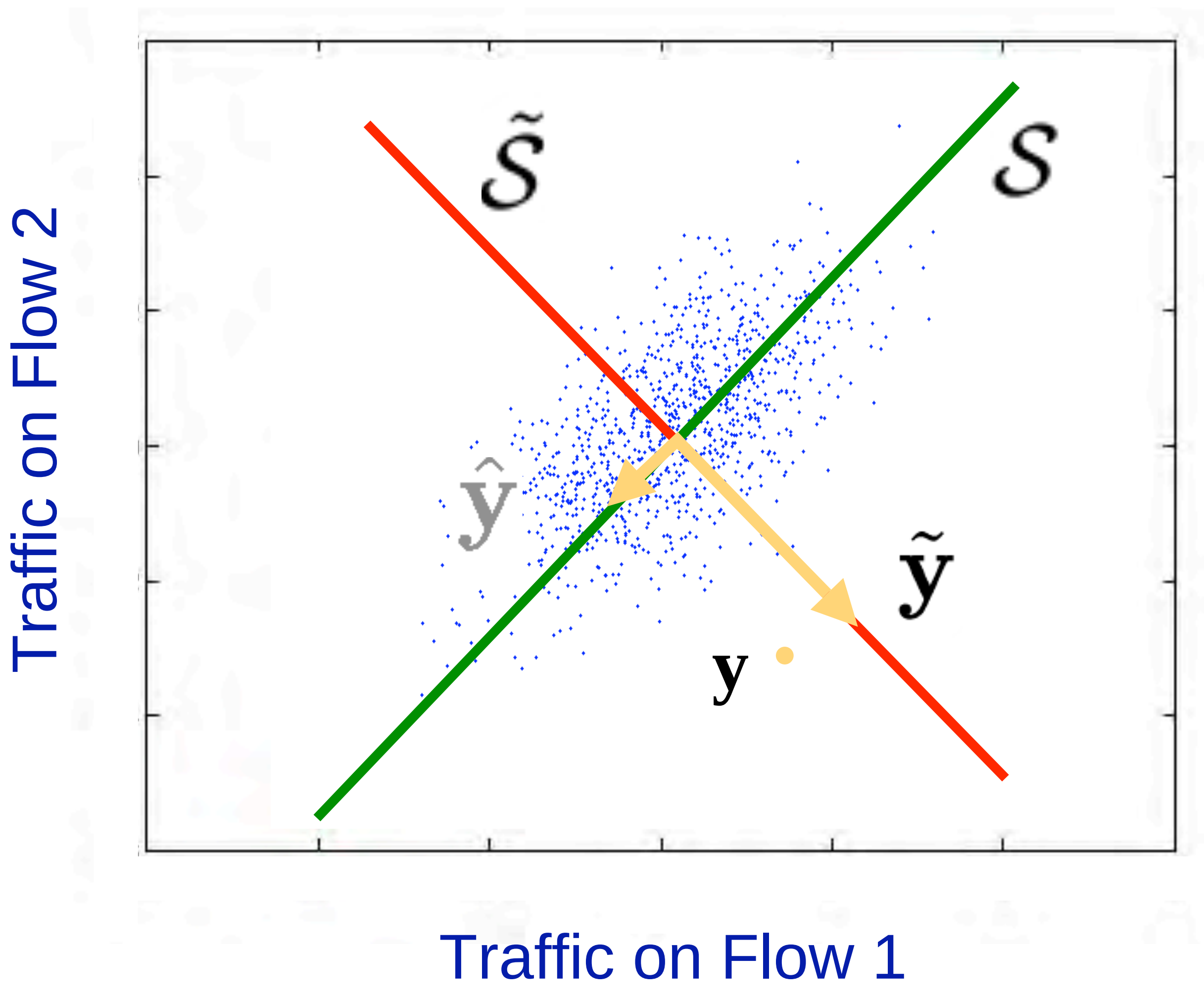
- An approach to separate normal & anomalous network-wide traffic
- Designate temporal patterns most common to all the OD flows as the **normal subspace**
- Remaining temporal patterns form the **anomalous subspace**
- Then, decompose traffic in all OD flows by *projecting* onto the two subspaces to obtain:

$$\underbrace{\mathbf{y}}_{\text{Traffic vector of all OD flows at a particular point in time}} = \underbrace{\hat{\mathbf{y}}}_{\text{Normal traffic vector}} + \underbrace{\tilde{\mathbf{y}}}_{\text{Residual traffic vector}}$$

Normal subspace

The Subspace Method, Geometrically

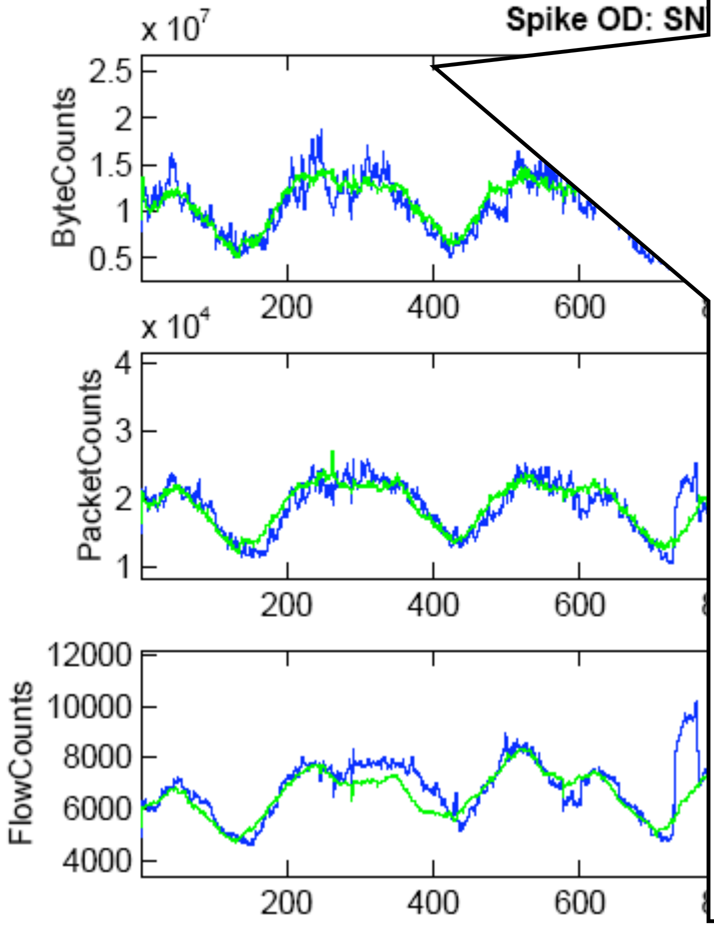
Anomalous subspace

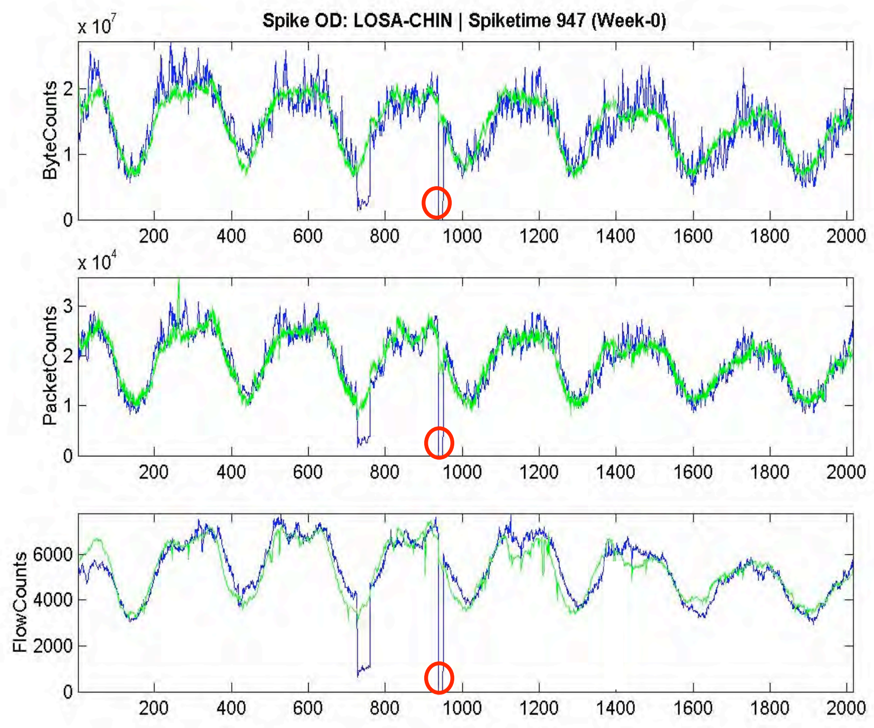


In general, anomalous traffic results in a large size of $\tilde{\mathbf{y}}$

For higher dimensions, use Principal Component Analysis
[LPC+:SIGMETRICS ‘04]

Example of a Volume Anomaly [LCD:IMC '04]





Multihomed customer CALREN reroutes around outage at LOSA

Talk Outline

- Methods
  - Measuring Network-Wide Traffic
  - Detecting Network-Wide Anomalies
  - **Beyond Volume Detection: *Traffic Features***
  - Automatic Classification of Anomalies
- Applications
  - General detection: scans, worms, flash, *etc.*
  - Detecting Distributed Attacks
- Summary

Exploiting Traffic Features

- **Key Idea:**

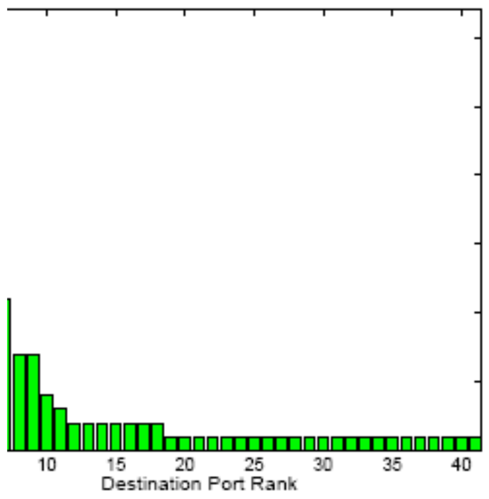
  Anomalies can be detected and distinguished by inspecting *traffic features*:
  `SrcIP, SrcPort, DstIP, DstPort`

- **Overview of Methodolgy:**

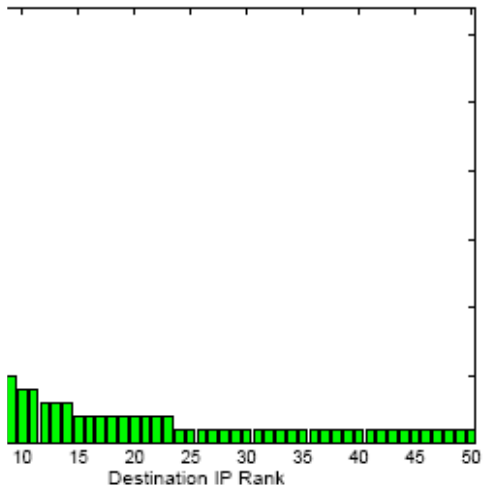
  1. Inspect *distributions* of traffic features
  2. Correlate distributions *network-wide* to detect anomalies
  3. *Cluster* on anomaly features to classify

Traffic Feature Distributions [LCD:SIGCOMM '05]

Dispersed Histogram
High Entropy



Concentrated Histogram
Low Entropy



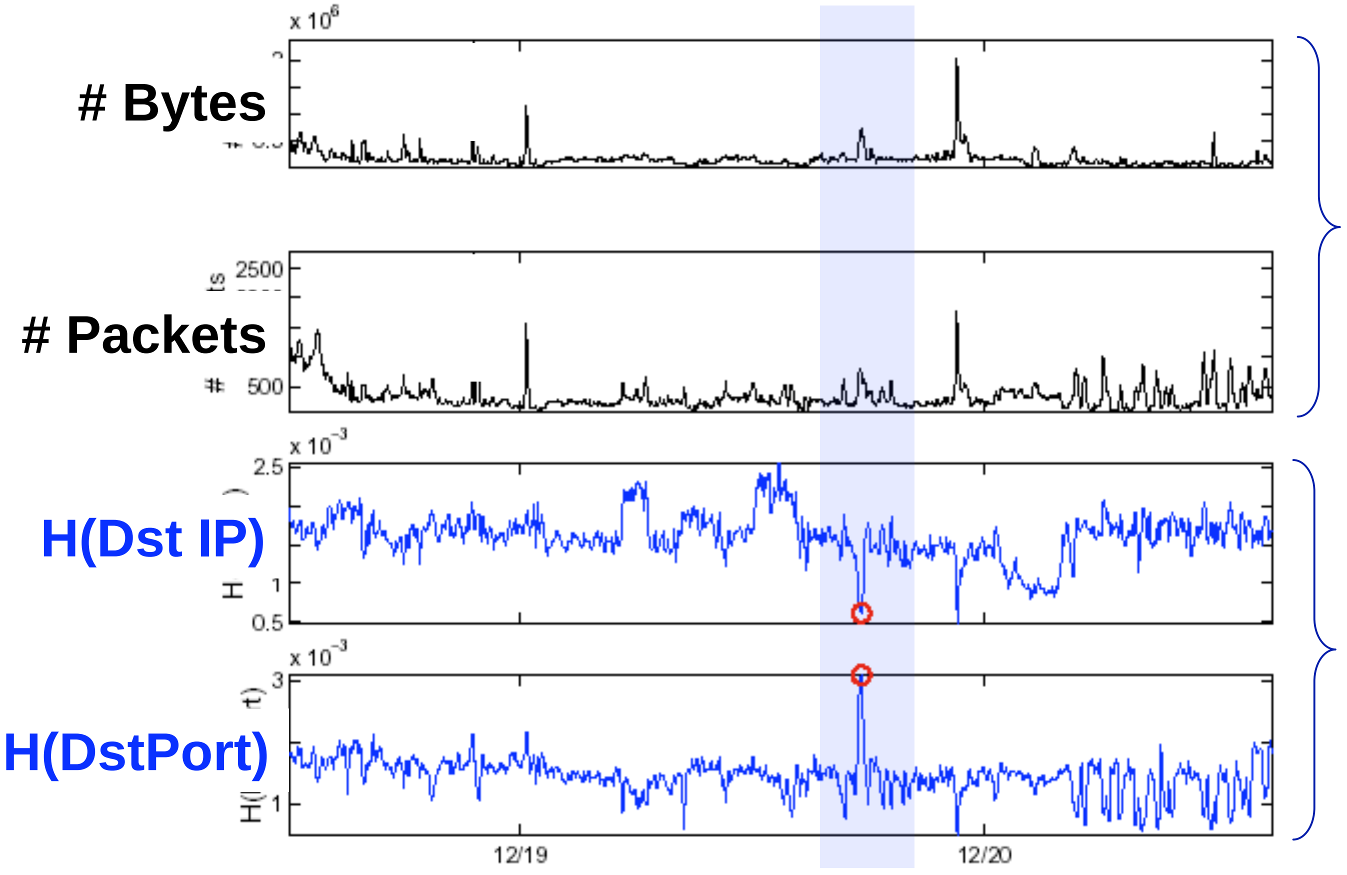
Typical Traffic

Summarize using **sample entropy** of histogram *X*:

$$H(X) = -\sum_{i=1}^{N} \left(\frac{n_i}{S}\right) \log_2 \left(\frac{n_i}{S}\right)$$

where symbol *i* occurs n_i times; *S* is total # of observations

Feature Entropy Timeseries



Port scan dwarfed in volume metrics…

But stands out in feature entropy, which also reveals its structure

How Do Detected Anomalies Differ?

Anomaly Label	# Found in Volume	# Additional in Entropy
Alpha	84	137
DOS	16	11
Flash Crowd	6	3
Port Scan	0	30
Network Scan	0	28
Outage	4	11
Point Multipoint	0	7
Unknown	19	45
False Alarm	23	20
Total	**152**	**292**

3 weeks of Abilene anomalies classified manually

Talk Outline

- Methods
  - Measuring Network-Wide Traffic
  - Detecting Network-Wide Anomalies
  - Beyond Volume Detection: *Traffic Features*
  - **Automatic Classification of Anomalies**
- Applications
  - General detection: scans, worms, flash events, …
  - Detecting Distributed Attacks
- Summary

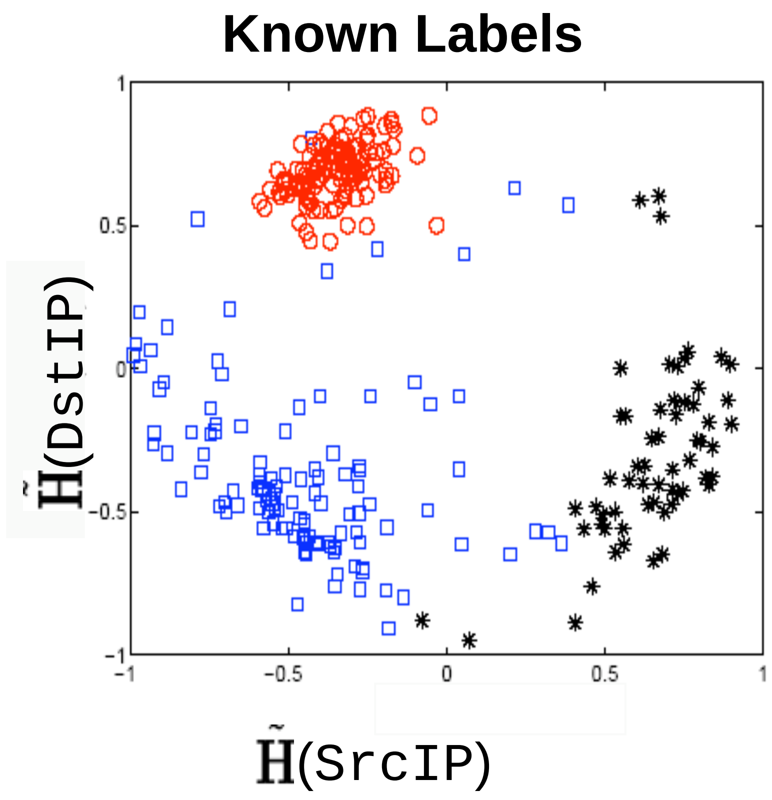
Classifying Anomalies by Clustering

- Enables unsupervised classification

- Each anomaly is a point in 4-D space:
  $[\ \tilde{H}(\texttt{SrcIP}),\ \tilde{H}(\texttt{SrcPort}),\ \tilde{H}(\texttt{DstIP}),\ \tilde{H}(\texttt{DstPort})\ ]$

- Questions:
  - Do anomalies form clusters in this space?
  - Are the clusters meaningful?
    - Internally consistent, externally distinct
  - What can we learn from the clusters?

Clustering Known Anomalies (2-D view)



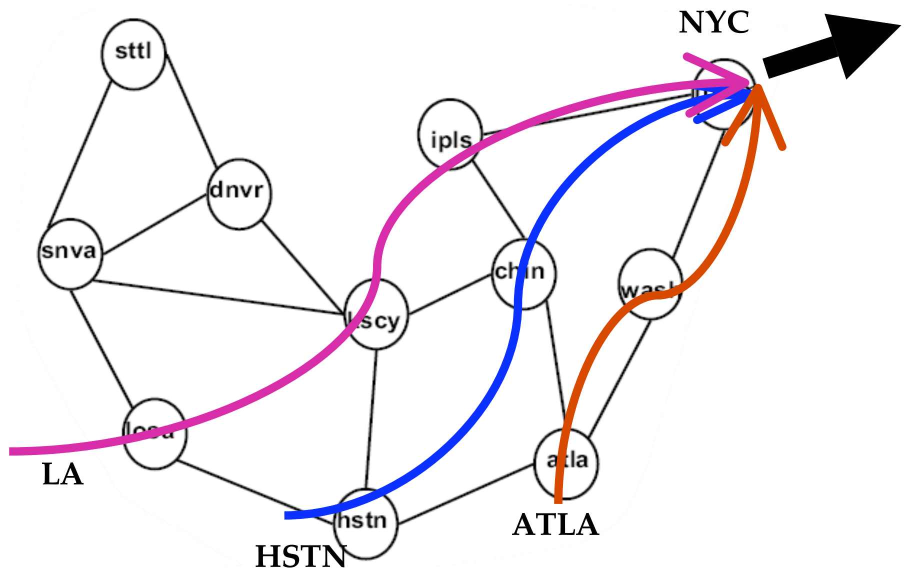
Legend

Code Red Scanning

Single source DOS attack

Multi source DOS attack

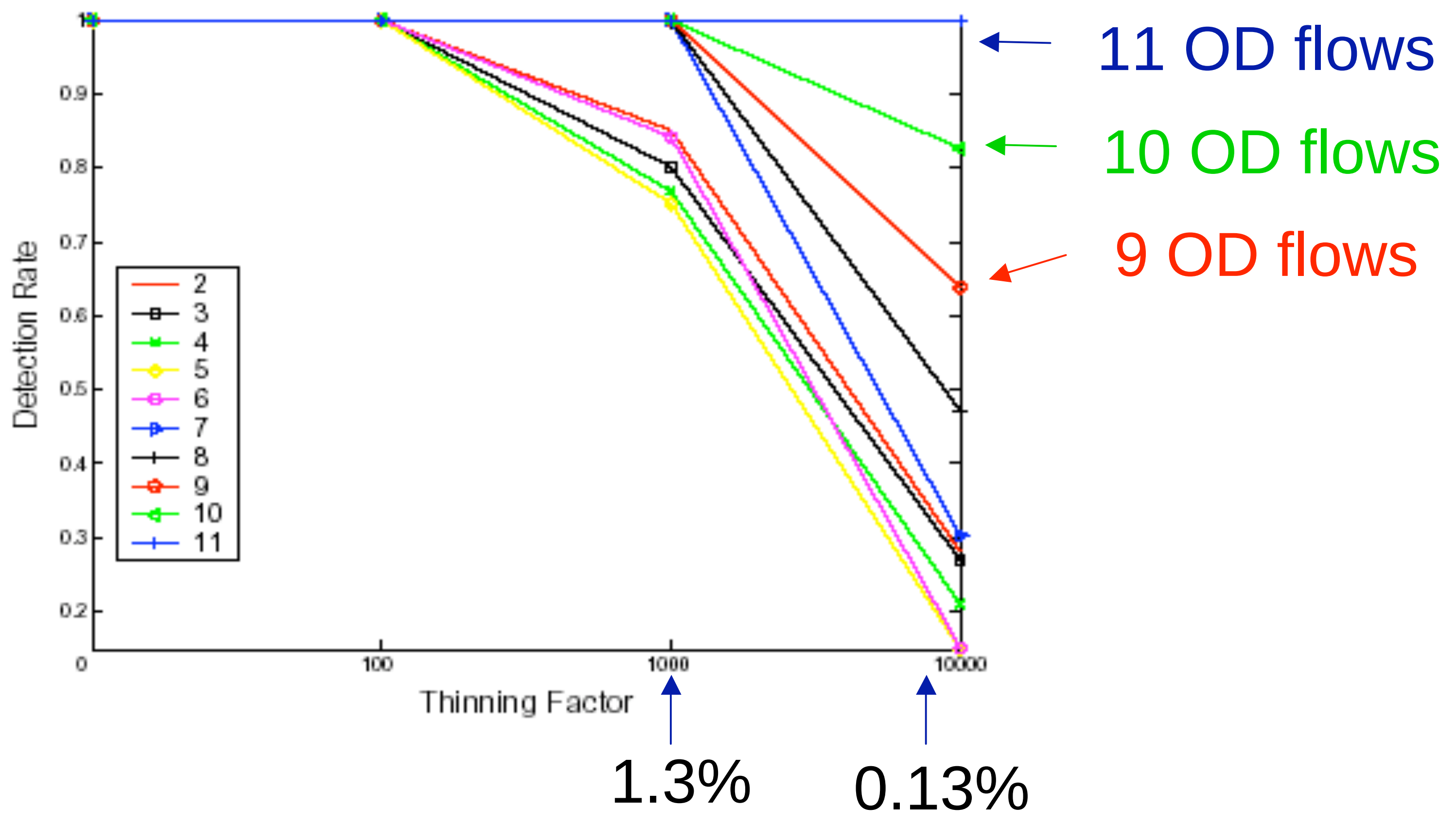
Back to Distributed Attacks…



Evaluation Methodology

1. Superimpose known DDOS attack trace in OD flows
2. Split attack traffic into varying number of OD flows
3. Test sensitivity at varying anomaly intensities, by thinning trace
4. Results are average over an exhaustive sequence of experiments

Distributed Attacks: Detection Results



The more distributed the attack, the easier it is to detect

Summary

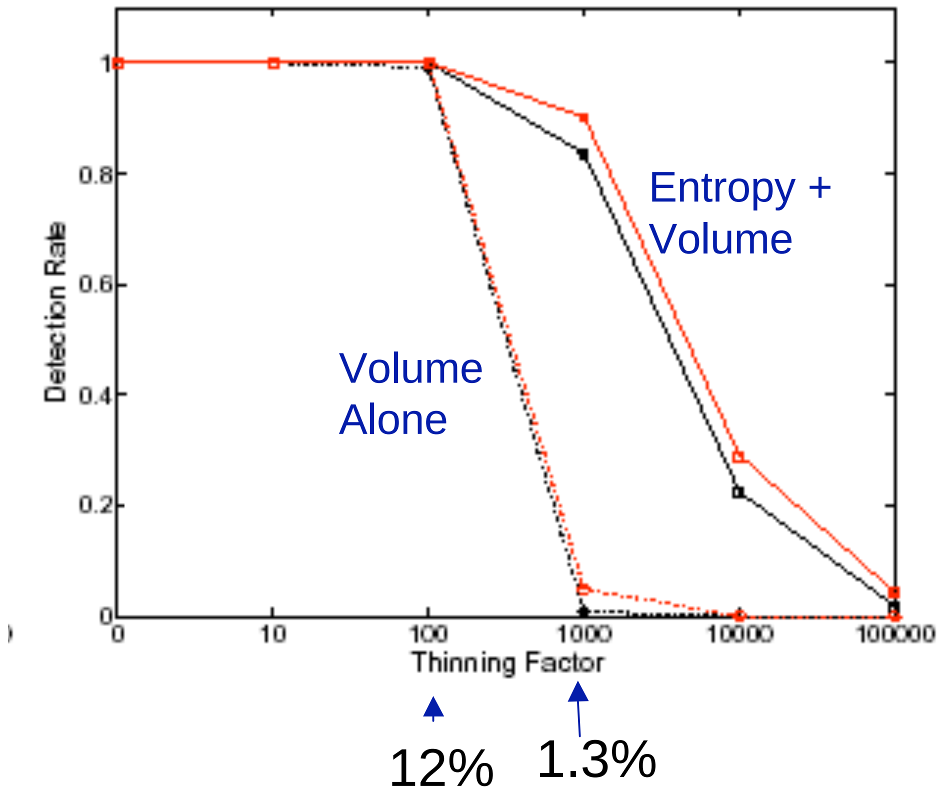
- Network-Wide Detection:
  - Broad range of anomalies with low false alarms
  - Feature entropy significantly augment volume metrics
  - Highly sensitive: Detection rates of 90% possible, even when anomaly is 1% of background traffic
- Anomaly Classification:
  - Clusters are meaningful, and reveal new anomalies
  - In papers: more discussion of clusters and Géant
- Whole-network analysis and traffic feature distributions are promising for general anomaly diagnosis

Backup Slides

Detection Rate by Injecting Real Anomalies

Multi-Source DOS

[Hussain et al, 03]



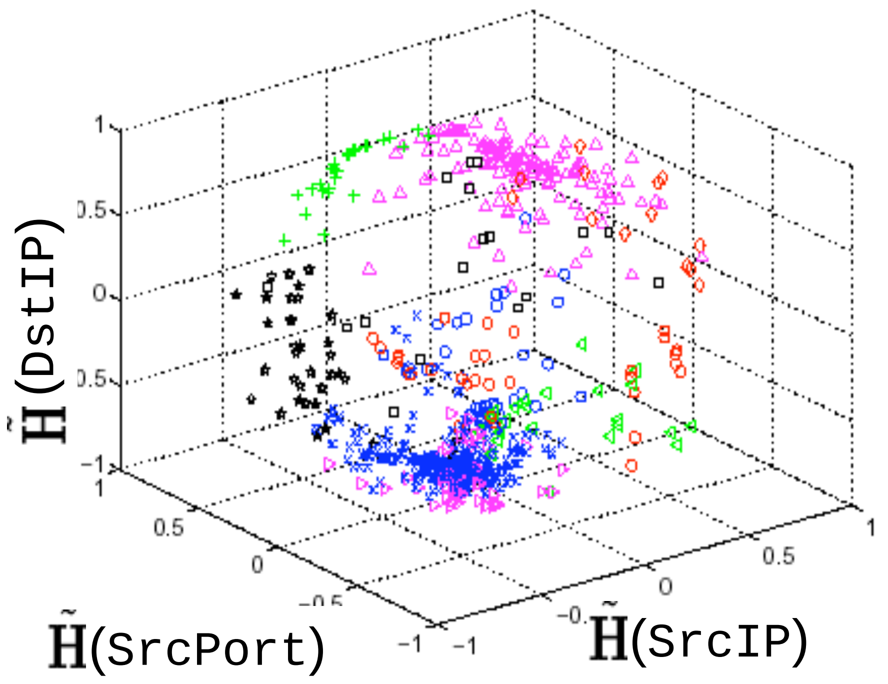
Detection rate vs. Anomaly intensity

(intensity % compared to average flow bytes)

Evaluation Methodology

- Superimpose known anomaly traces into OD flows
- Test sensitivity at varying anomaly intensities, by thinning trace
- Results are average over a sequence of experiments

3-D view of Abilene anomaly clusters



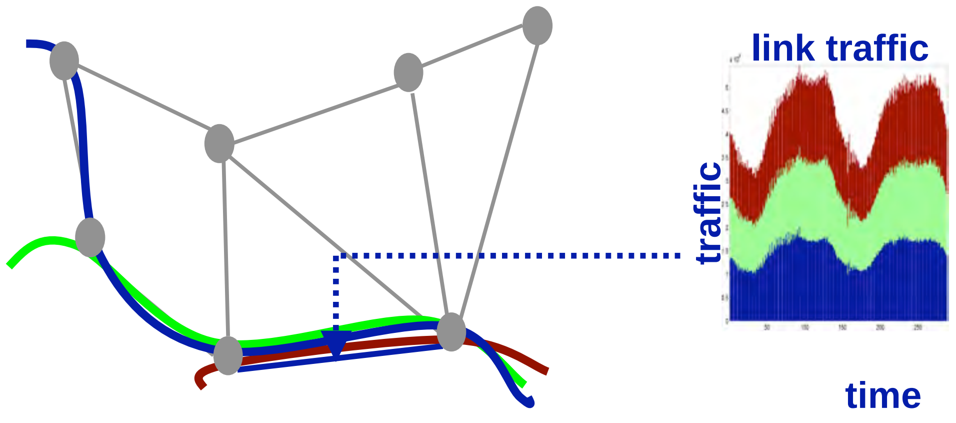
- Used 2 different clustering algorithms
  - Results consistent

- Heuristics identify about 10 clusters in dataset
  - details in paper

Anomaly Clusters in Abilene data

ID	# points	Plurality Label	$\tilde{\mathbf{H}}(\text{srcIP})$	$\tilde{\mathbf{H}}(\text{srcPort})$	$\tilde{\mathbf{H}}(\text{dstIP})$	$\tilde{\mathbf{H}}(\text{dstPort})$
1	191	**Alpha**	–	0	–	–
2	53	**Network Scan**	0	+	0	0
3	35	**Port Scan**	–	+	–	+
4	30	**Port Scan**	0	–	0	+
5	24	**Alpha**	0	0	+	0
6	22	**Outage**	0	0	0	+
7	22	**Alpha**	–	0	–	0
8	8	**Point Multipoint**	0	0	0	+
9	8	**Flash Crowd**	0	0	0	–
10	4	**Alpha**	0	–	0	0

Why Origin-Destination Flows?



- All link traffic arises from the superposition of OD flows
- OD flows capture *distinct* traffic demands; no redundant traffic
- A useful primitive for whole-network analysis

Subspace Method: Detection

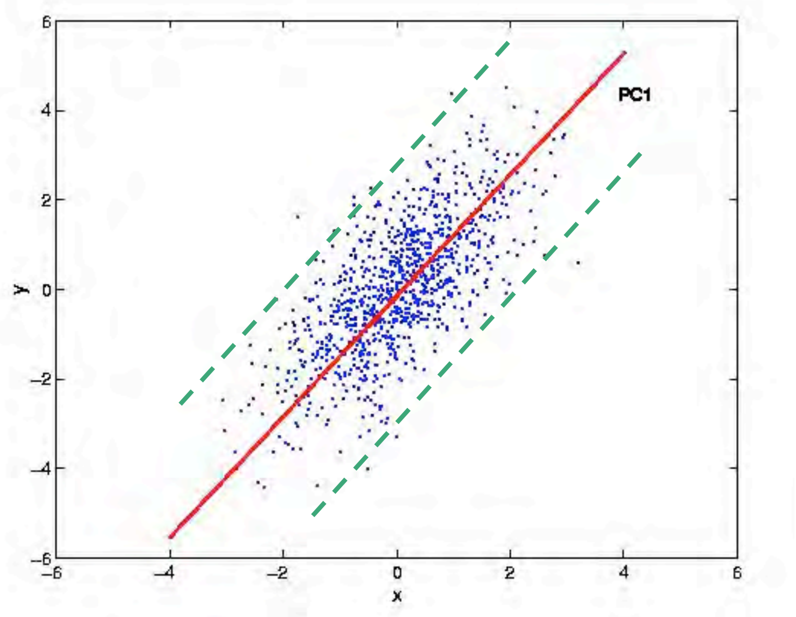
- Error Bounds on Squared Prediction Error:

$$\mathrm{SPE} \equiv \|\tilde{\mathbf{y}}\|^2 = \|\tilde{\mathbf{C}}\mathbf{y}\|^2$$

- Assuming Normal Errors:

$$\mathrm{SPE} \leq \delta_\alpha^2$$

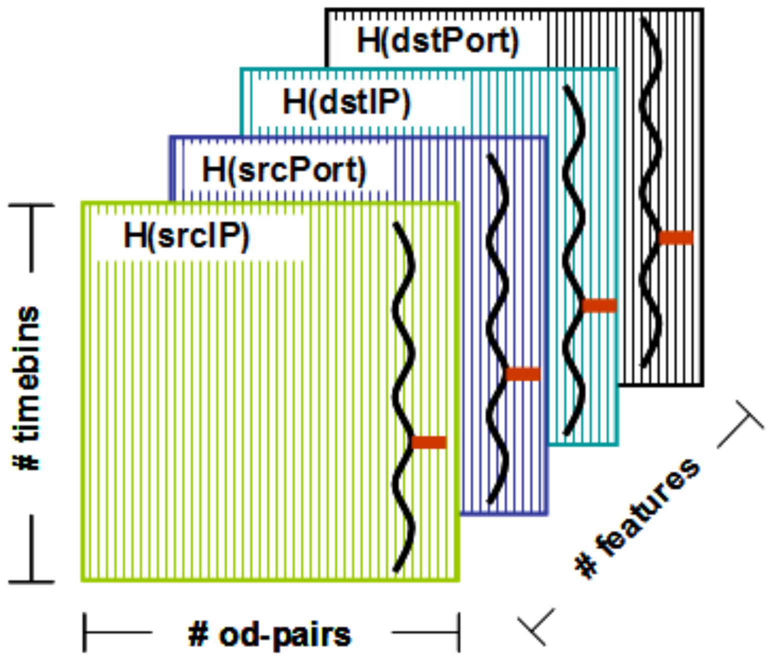
- Result due to
  [Jackson and Mudholkar, 1979]



Subspace Method: Identification

- An anomaly results in a displacement of the state vector away from $\mathcal{S}$
- The direction of the displacement gives information about the nature of the anomaly
- Intuition: find the OD flow that best describes the direction associated with a detected anomaly
- More precisely, we select the OD flow that accounts for maximum residual traffic

Network-Wide Traffic Data Collected

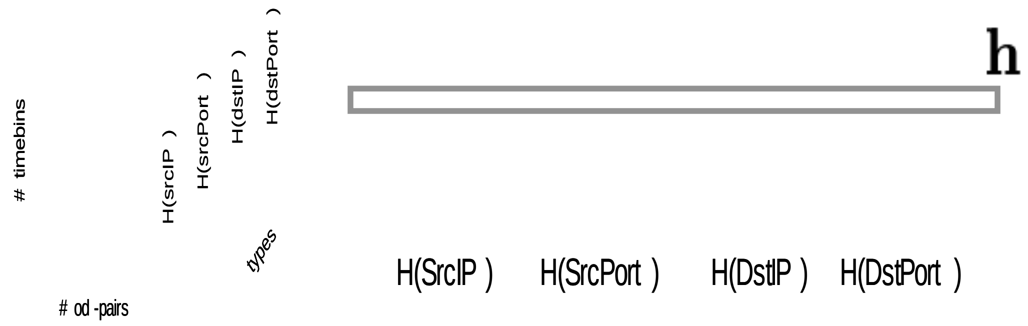


Multivariate, multiway timeseries to analyze

- Compute entropy on packet histograms for 4 traffic features: **`SrcIP, SrcPort, DstIP, DstPort`**

Multiway Subspace Method

1. “Unwrap” the multiway matrix into one matrix



Time, Pollution and Maps

Michael Collins, CERT/NetSA

Software Engineering Institute

How’re we doing?

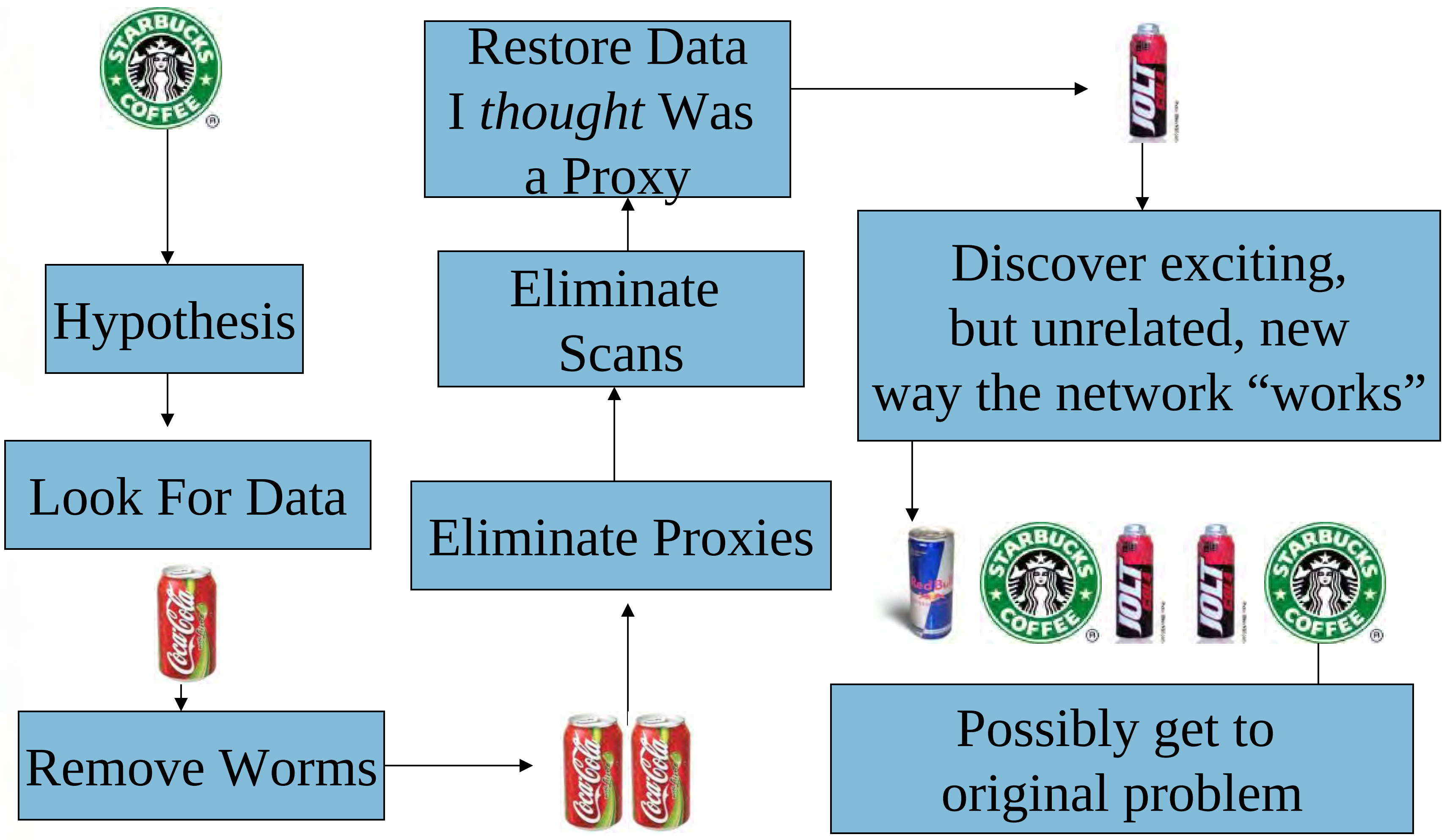
The basic cost: time

- Time to analyze
- Time to verify
- Time to retrack when we make mistakes

Basic success:

- In time t, x *things* happen
  – We understand > *x* in time *t*: good!
  – We understand < *x* in time *t:* bad!
- We’re probably at <<x right now

My (*work*) flow


Hypothesis
Look For Data
Remove Worms
Eliminate Proxies
Eliminate Scans
Restore Data I *thought* Was a Proxy
Discover exciting, but unrelated, new way the network "works"
Possibly get to original problem

Why Flow?

Ultimate cost: time

- Time = (storage) space

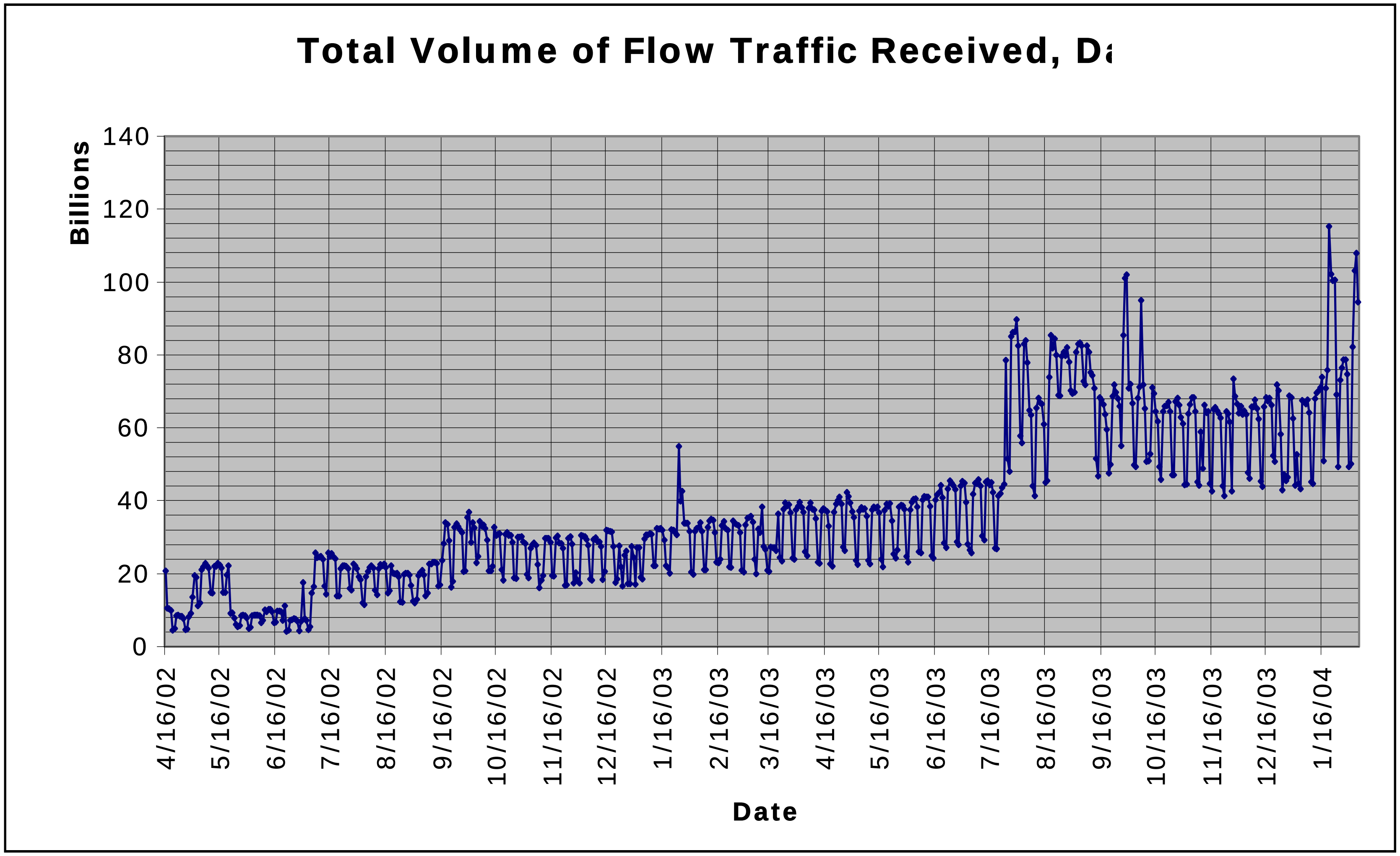
Basic issue - bang for the buck

- Catastrophe - the internet is regularly reconfigured, traffic volumes suddenly shift
- Pollution - approximately 70-80% of the TCP flows we see are not legitimate sessions

Flow is manageable where pure payload generally isn't

- I am looking at effectively *random* collections of packets
- Flow is the highest value information from a random collection of packets

Still have a basic problem



Manageable Additions

Adding additional flow information costs us

- Expression = field size = performance
- Additional data on disk should allows us to understand more *things*

Certain additions are going to come whether we like it or not

- IPv6
- Sasser

Expanding Flow Analysis

Fundamental Goal: *What's up?*

Secondary Goal: Don't break the bank

- Context
- Grouping
- Expansion

Context

Preserving knowledge of what’s *on* the network

- Trickler
- Mapping - DNS, BGP, ICMP, etc.

Shouldn’t have to repeatedly do ad hoc discovery

Maps *should* be smaller

Grouping

Annotating multiple flows together as one event

- Scan detection
- BitTorrent Distribution
- Websurfing

Don’t reconstruct this on a per-query basis

Expansion

Expand to *increase distinguishability*

- Increased time precision
- Some payload information

Try not to expand in order to identify *specific things*

- We *will* be attacked, any specific attack *implementation* is therefore of limited value

Concrete Suggestions

Heterogenous Splits:

- Full ICMP
- Short events
- Characteristics of payload
- Protocol validation

Conclusions

Our primary currency is time

- Time to access
- Time for backtracking
- Time for figuring out what the heck is going on

Time is equivalent to space

- Data on disk governs how long it takes to read information
- 10 billion events/day is about 2 DVDs/byte

Behavior Based Approach to Network Traffic Analysis

Rob Nelson

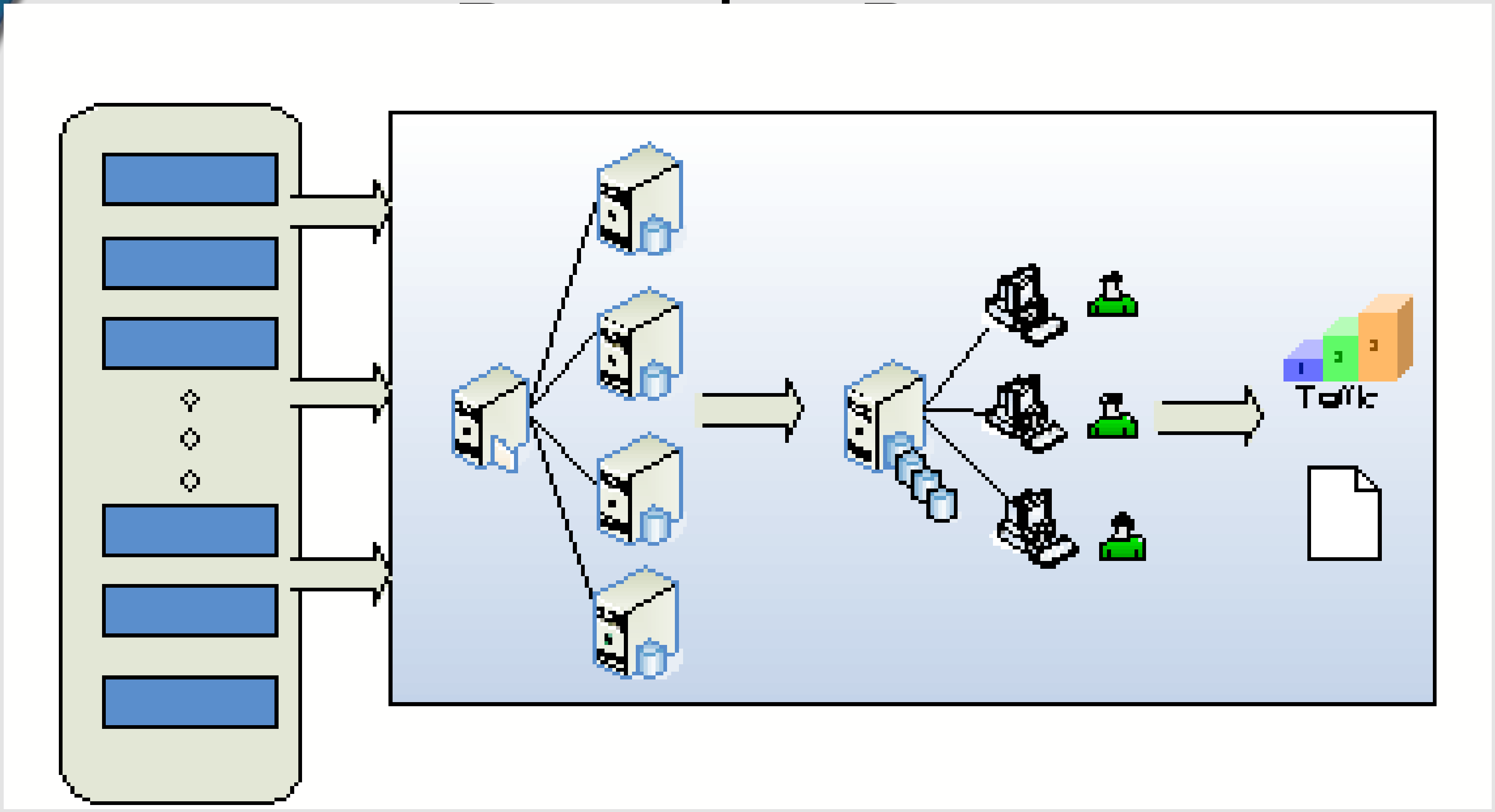
Casey O’Leary

Pacific Northwest National Laboratory

Issues/Challenges

- Data volume (noisy/highly dimensional)
- Watch-lists
- Data interpretation – significance
- Monitoring threats

Data Collation, Processing, Analysis, and



Advancing the Art

- Situation awareness
  - Recognize nefarious activities before reported
  - Focusing analysts on particular IP’s or organizations
- Novel analysis
  - Identifying exploits before well known

Dynamic Watch-lists

- External hosts
  - Those IP addresses that pose a threat against the enterprise networks
- Vulnerable hosts
  - Those internal IP address that are targets

Methods

- Weighted values associated with behavior
- Tracking over time
- Dynamic list placement
- Behavior profiles
- Multiple sources of input

External Hosts

- Actions
  - Reconnaissance
  - Exploits
- Intent
  - Collection
  - Compromise
  - Recruitment
- Methods
  - Stealth
  - Collaboration

Vulnerable Hosts

- Interacting with external hosts
- Sending unsolicited messages
- High level of chatter
- New services running

Factors

- Intent
- Temporal/frequency
- Sophistication
- Cooperation
- Enclave

Adaptability

- Dynamic weighting factors
- Methods
- Techniques
- Code

Future Efforts

- Architecture to work on raw data
  - Near real-time situation awareness
  - Parallelism of queries
- Sophisticated detection methods
- Communities

Questions

- Contacts

Rob Nelson	Casey O’Leary
rob.nelson@pnl.gov	*casey@pnl.gov*

FloCon
2005 ANALYSIS WORKSHOP

CERT

R: A Proposed Analysis and Visualization Environment for Network Security Data

Josh McNutt <jmcnutt@cert.org>

Software Engineering Institute

Outline

SiLK Tools

Analyst’s Desktop

Introduction to R

R-SiLK Library (Proof-of-concept prototype)

Context Objects and Analysis Objects

Analyst Benefits

Prototype Demo

Future of Analyst’s Desktop

SiLK Tools

System for Internet-Level Knowledge

- http://silktools.sourceforge.net/

Developed and maintained by CERT/NetSA (Network Situational Awareness) Team

Consists of a suite of tools which collect and perform analysis operations on NetFlow data

Optimized for very large volume networks

Command Line Interface (CLI)

Fundamental Tools

- rwfilter
- rwcount
- rwuniq
- IP sets

Enhancing SiLK: Analyst's Desktop

We are currently developing a new interface model for the SiLK tools

The goal is to develop an environment supporting sophisticated analysis of network security phenomena

- **Analyst's Desktop**

Requirements

- Interactive visualization capability
- Audit trail
- Annotation
- Preserve the command line options available in SiLK
- Make simple analyses simple to perform

Platform of choice: **R**

R: What is it?

R is a programming language and environment for statistical computing and graphics used by statisticians worldwide

The R Project for Statistical Computing

- http://www.r-project.org

R is available as Free Software under the terms of the Free Software Foundation's GNU General Public License in source code form

There exists a thriving community of statisticians and statistical programmers who contribute their code

R! What is it good for?

R represents “best-in-practice” environment for exploratory data analysis

Specifically designed with data analysis in mind

- A more natural analysis interface than Perl, Python or shell scripts

Full Access to R’s built-in statistical analysis capability

R can run interactively or in batch mode

Visualization

- Integrated graphing capabilities (publication quality)

R! What is it good for?

Object-based environment

- Everything in R is an object
  - – functions, matrices, vectors, arrays, lists
- Objects can be saved in user workspace (persistence) or saved to disk and sent to another user's workspace
- Preserve results for comparison with future analyses
- Annotations can be attached to objects

Command line control can be preserved

- Wrapper functions incorporate SiLK command line arguments

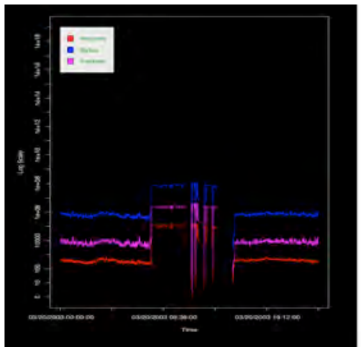
Rapid prototyping of new analysis techniques and visualizations

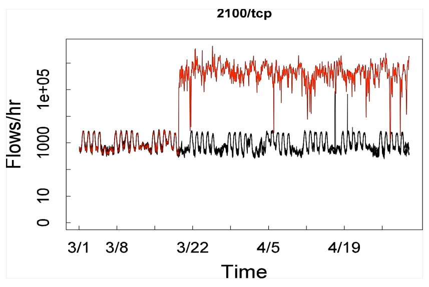
R's Graphing Capability

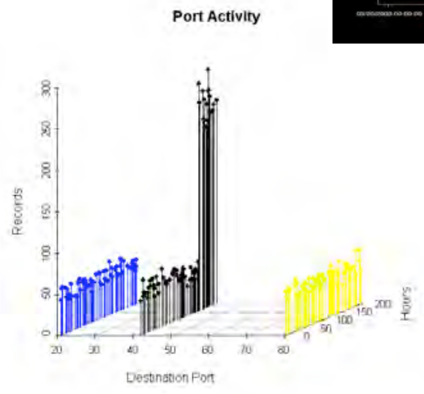
Huge set of standard statistical graphs

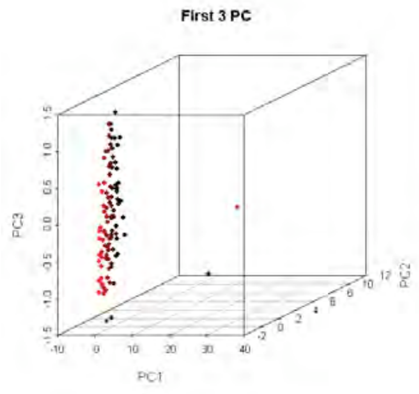
- stemplots, boxplots, scatterplots, etc.

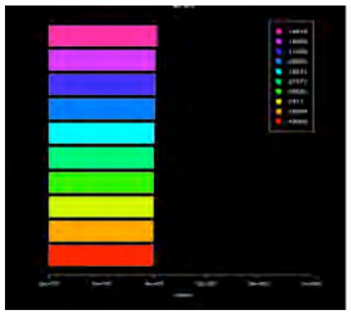
3D graphing capability

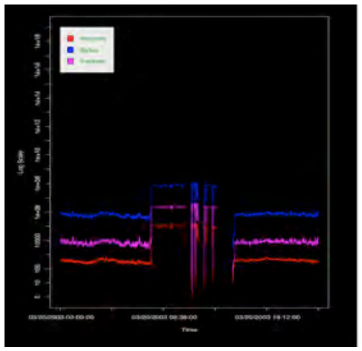












R-SiLK Library

Low-level interface involves custom wrapper functions making command line calls to SiLK

Higher-level functions call those wrappers

Many SiLK Tools have associated functions in R-SiLK library

- rwfilter(), rwcount(), rwuniq(), rwcut()

The SiLK Tool “rwcount” generates a binned time series of records, bytes and packets

In R-SiLK library, there is a function called “rwcount()”

- rwcount(rwcount switches, context object)
- Example below assigns 60-second binned time series data for context object called “context.tcp” to the analysis object called “analysis.tcp”
    - analysis.tcp <- rwcount(“--bin-size=60”, context.tcp)

Context objects and analysis objects?

Context Objects and Analysis Objects

To aid in analysis tasks, we've created something called a context object

Context Object

- An object in R that determines precisely what data is being considered
  - Contains a text string element indicating filter criteria (**rwfilter switches**)
  - Contains the name of the binary file of flow data satisfying the filter criteria
- Simple example (Time period is only filter criteria)
    - All flows between midnight and 1 a.m. on July 17th, 2005
- Advanced example (Many additional criteria)
    - All inbound flows from source XXX.YYY.XXX.ZZ between midnight and 1 a.m. on July 17th, 2005 targeting any hosts in XXX.ZZZ.0.0/16 on destination port 42/tcp

As the analysts learn more about a particular context through analysis, they will be able to refine the current context by adding additional filter criteria

Context Objects and Analysis Objects

Context objects can be summarized/described via the process of analysis

To store the results of analysis we have analysis objects

Analysis Object

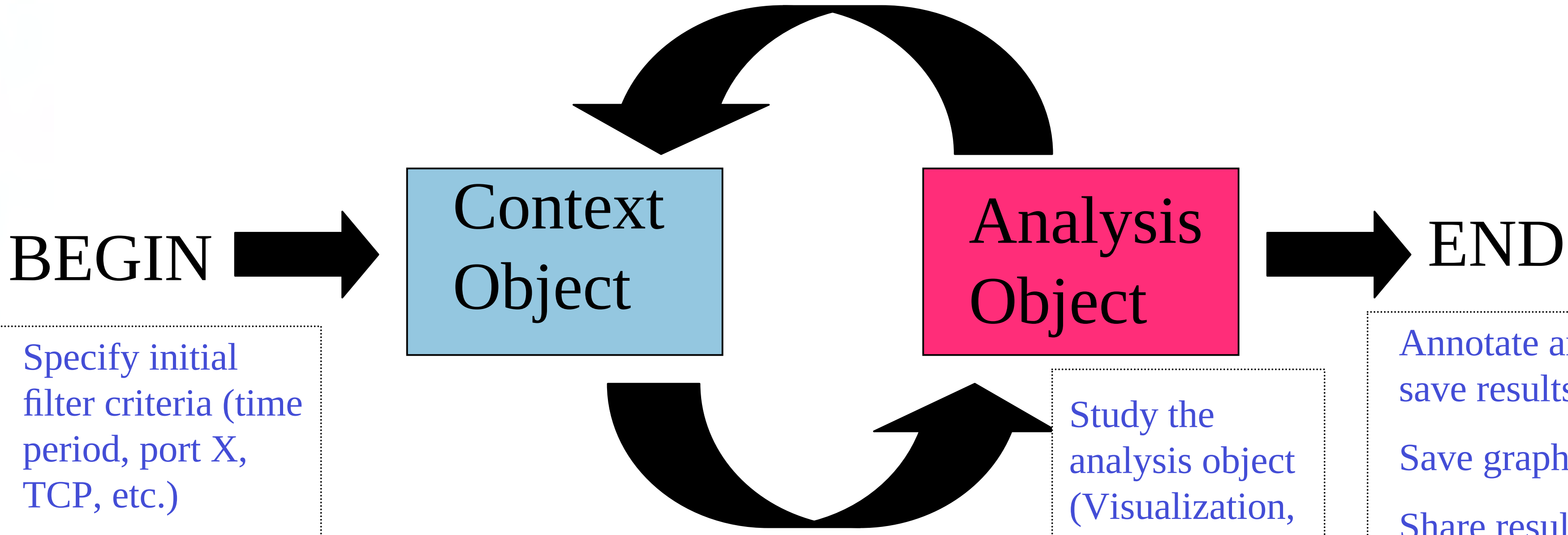
- An object in R that saves a description of a context object
  - Examples:
    - A top N list of destination ports for a context object
    - A binned time series of the flow data for a context object
- Components
  - Data (time series, sorted list of port volumes, etc.)
  - Context Object (what was the source data)
  - Timestamp (when was it created)
  - Descriptive Results (correlation, mean, etc.)
- Annotation
  - Can be attached to analysis object by analyst
  - Examples:
    - “UDP-based DDoS began around 8:30 a.m. on 5/6/04”
    - “Scanning appears to be targeting 2 local subnets”

Context Objects and Analysis Objects

Analysis Workflow

Refine Context (specify host, subnet, port number, etc.)

CONTEXT MENU (rw.analyze module)



Specify initial filter criteria (time period, port X, TCP, etc.)

Select an analysis

ANALYSIS MENU (rw.analyze module)

Study the analysis object (Visualization, Summary Stats, etc.)

Annotate and save results

Save graphs

Share results with other analysts

Analyst Benefits

Experienced Analyst

- Enhanced command line experience
  - Immediate and integrated visualization
- Object Persistence
- Annotation
- Audit Trail
- Rapid Prototyping

Beginner Analyst

- Faster time to productive investigations
- rw switches can be made transparent to the user
  - concatenated together in the background
- rw.analyze() module

Prototype Demo

R interactive mode

Basic proof-of-concept interface: rw.analyze()

Demonstrate the *Context Object – Analysis Object* workflow

Begin Demo

Future of Analyst’s Desktop

Working on improved version of R-SiLK library and prototype interface

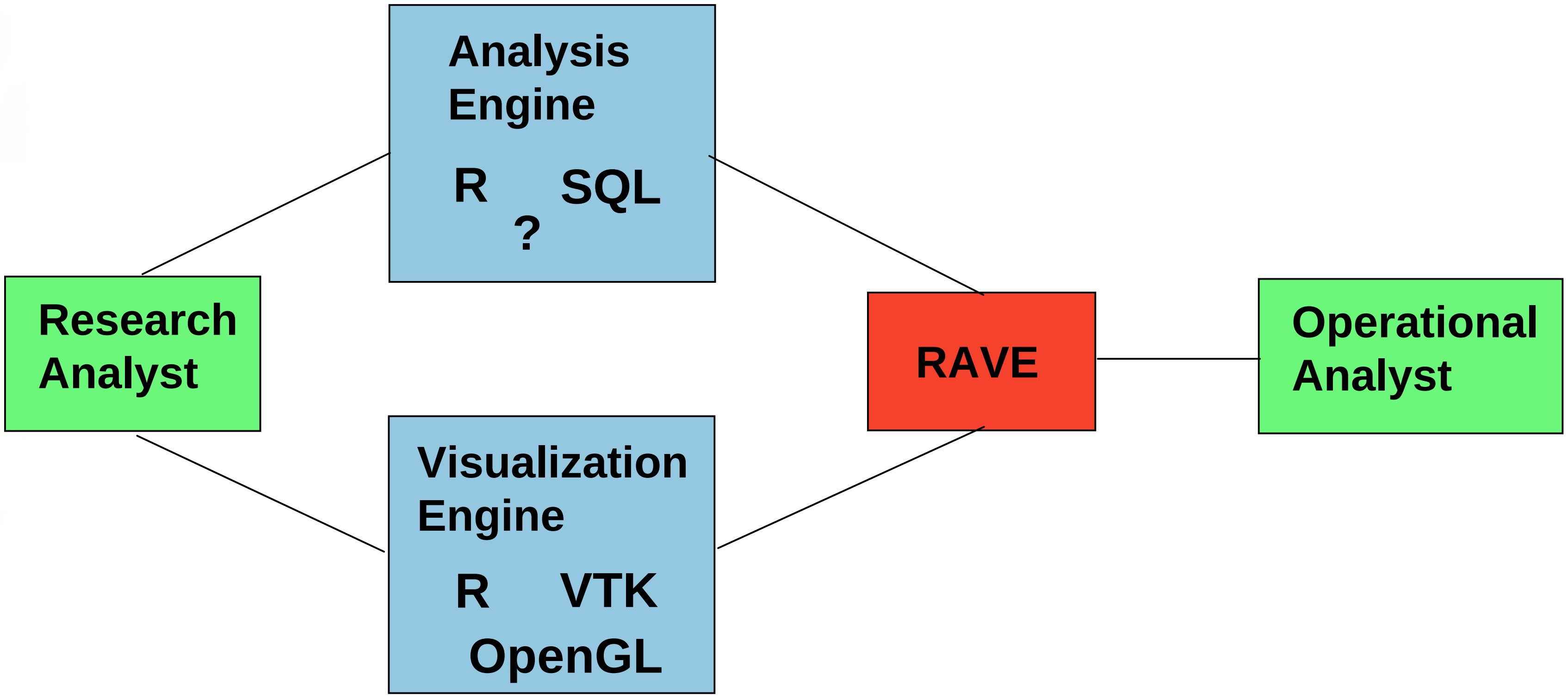
Support different modes of analysis

- Research Analysis
  - Flexible, powerful, customized
- Operational Analysis
  - Immediate, concise, “canned”

Future of Analyst's Desktop

RAVE

- Retrospective Analysis and Visualization Engine



Future of Analyst’s Desktop

RAVE

- Operationalize analysis techniques
  - Move new research techniques efficiently into operations
  - Furnish operational services (e.g. caching)
- Decouple analysis/visualization from UI
  - Different A/V tools, same UI
    - SiLK, R, SQL, Python/C, etc.
  - Different UIs, same engine
    - "Dashboards"
    - Menu of "canned" queries
    - Sophisticated data exploration environment (e.g., R)

Questions

QoSient LLC
Measuring Assessing and Managing QoS

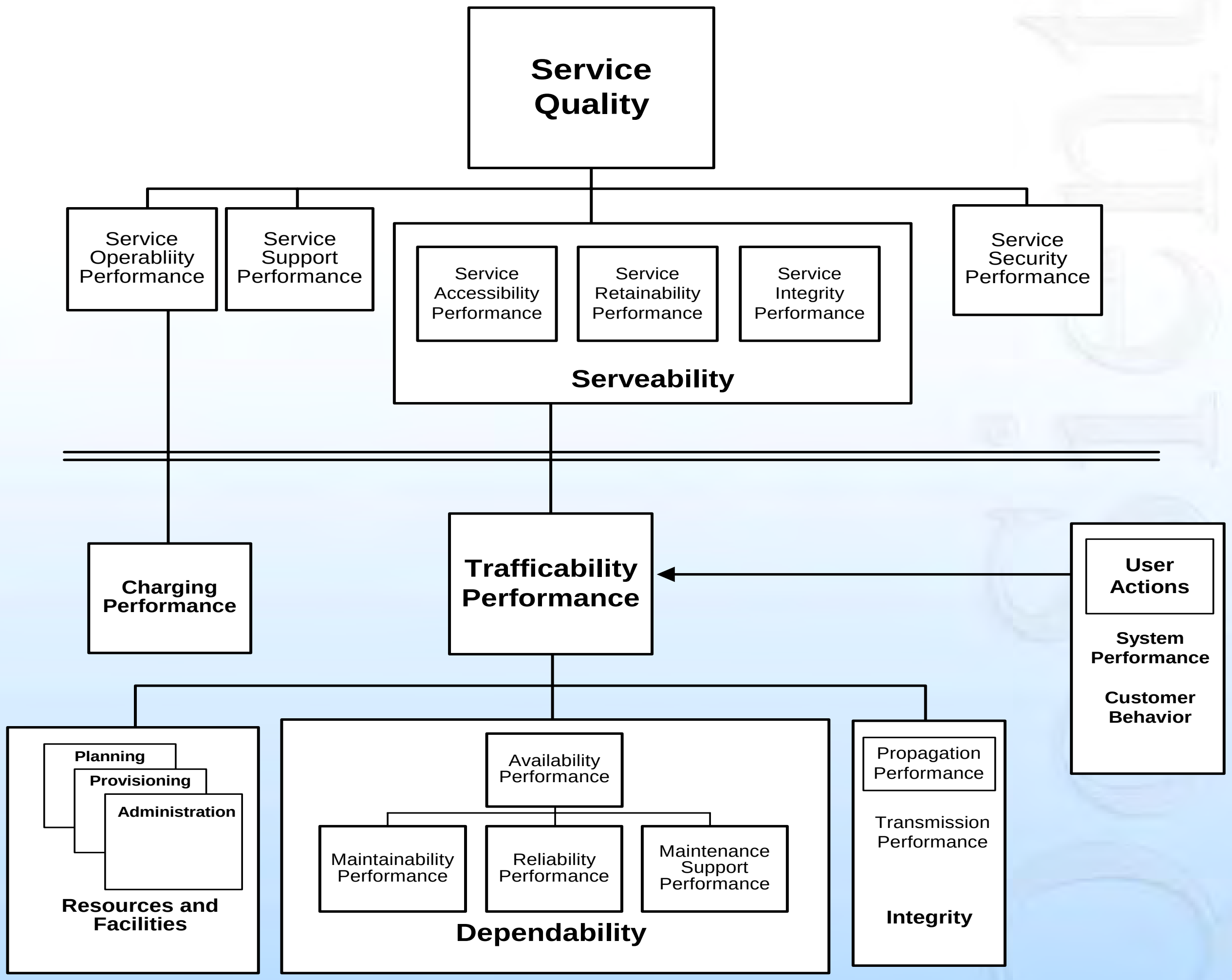
GIG^EF
Global Information Grid
Evaluation Facilities

Distributed QoS Monitoring
High Performance Network Assurance

Carter Bullard

FloCon 2005 Pittsburgh, PA

ITU Network Service Quality Taxonomy



10 October 2005

From ITU-T Recommendation E.800 Quality of Service, Network Management and Traffic Engineering

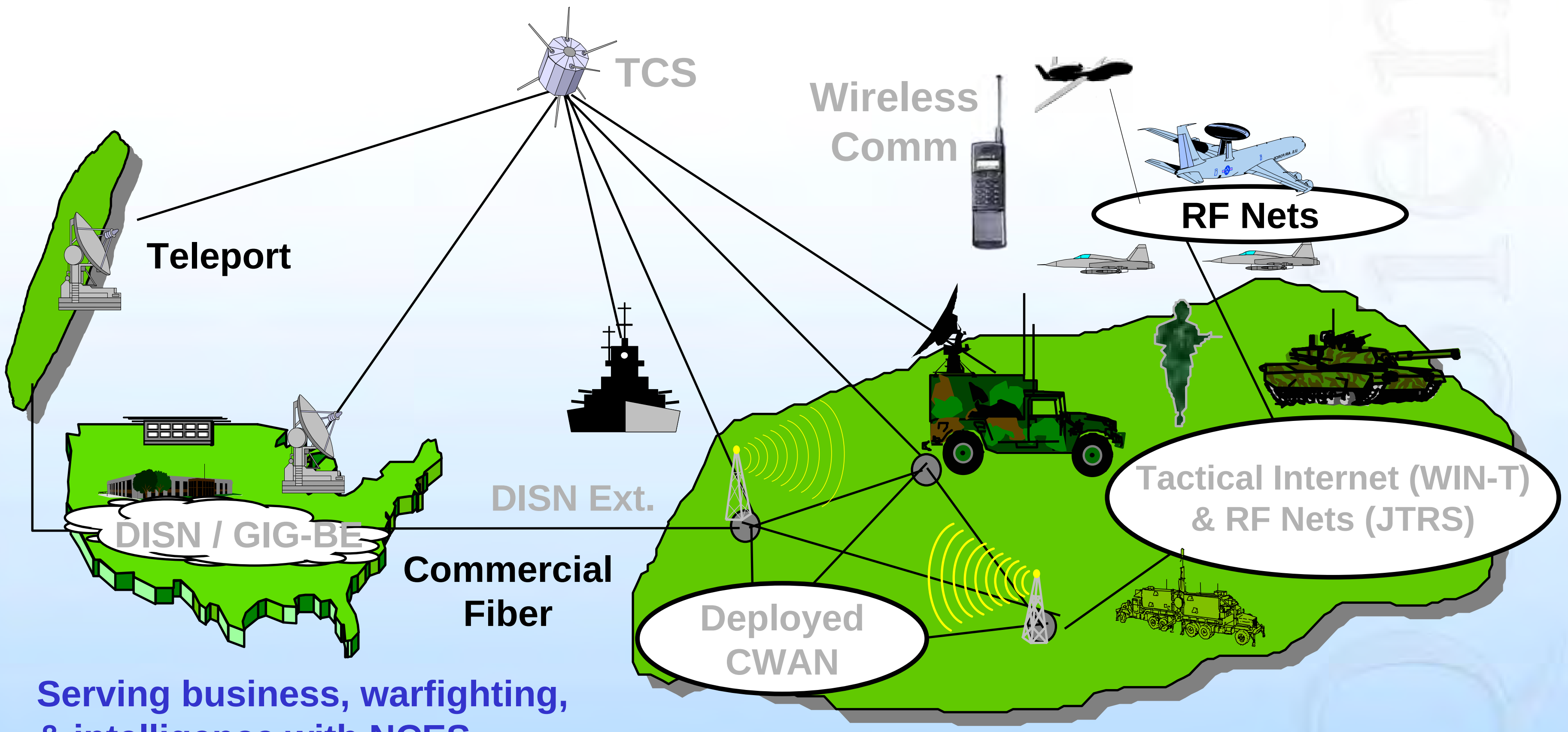
Approach

- Adopt PSTN TMN Usage Strategies
  - Service Oriented Metering
  - Integrated Measurement
  - Establish Comprehensive Transactional Audit
  - Near Real-Time Accessibility
- Extend PSTN Model for Internet Networking
  - Internet Transactional Model
  - Distributed Asymmetric Network Monitoring

Comprehensive Data Network Accountability

- Ability to account for all/any network use
- At a level of abstraction that is useful
  - Network Service Functional Assurance
    - Was the network service available?
    - Was the service request appropriate?
    - Did the traffic come and go appropriately?
    - Did it get the treatment it was suppose to receive?
    - Did the service initiate and terminate in a normal manner?
  - Network Control Assurance
    - Is network control plane operational?
    - Was the last network shift initiated by the control plane?
    - Has the routing service converged?

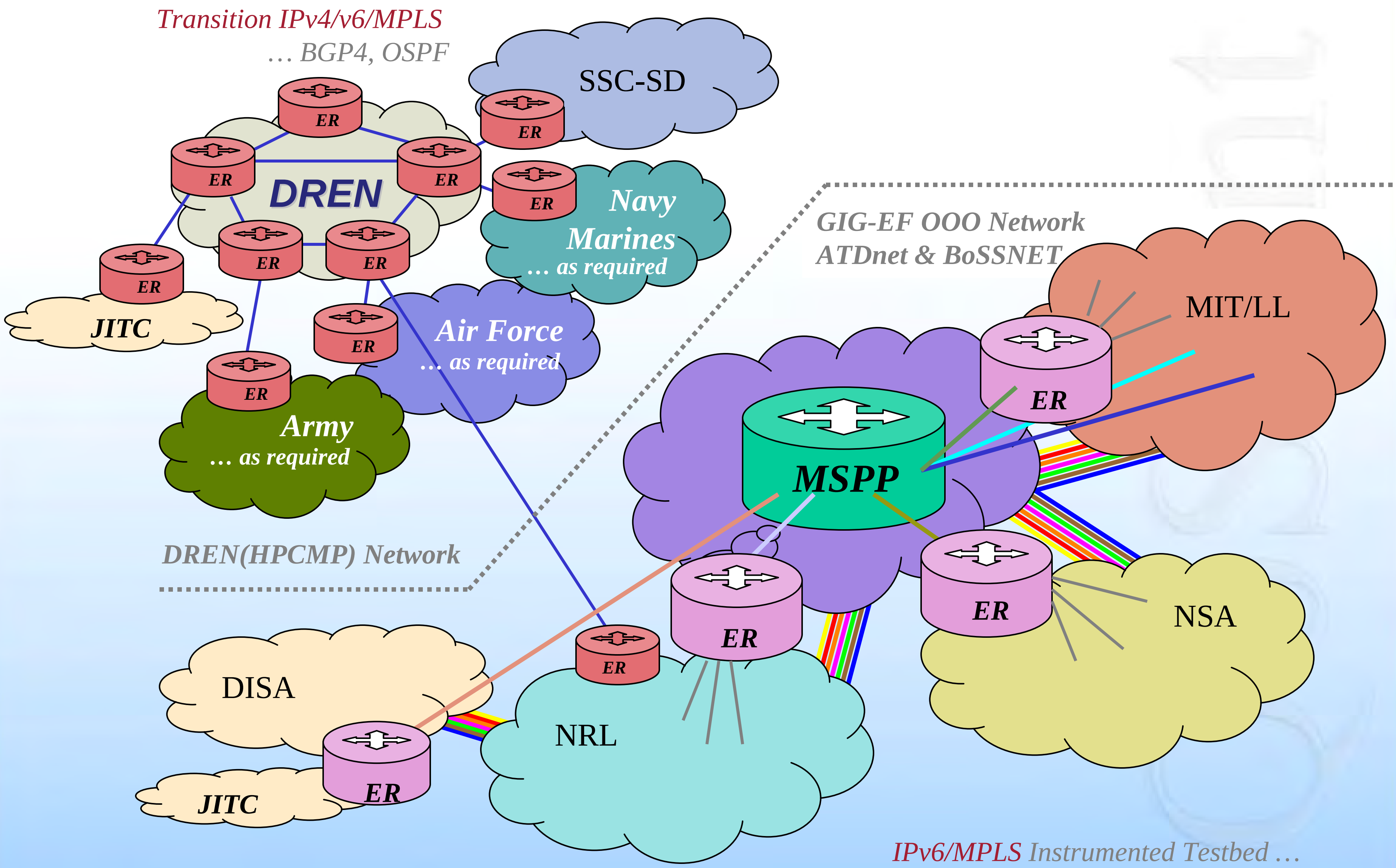
The Global Information Grid
A Diverse Environment



Serving business, warfighting, & intelligence with NCES --

- Collaboration, messaging, & applications
- Storage and mediation
- User assistance
- Information Assurance
- Enterprise Services Management and Operations

10 October 2005



10 October 2005

Abstract QoS Control Plane

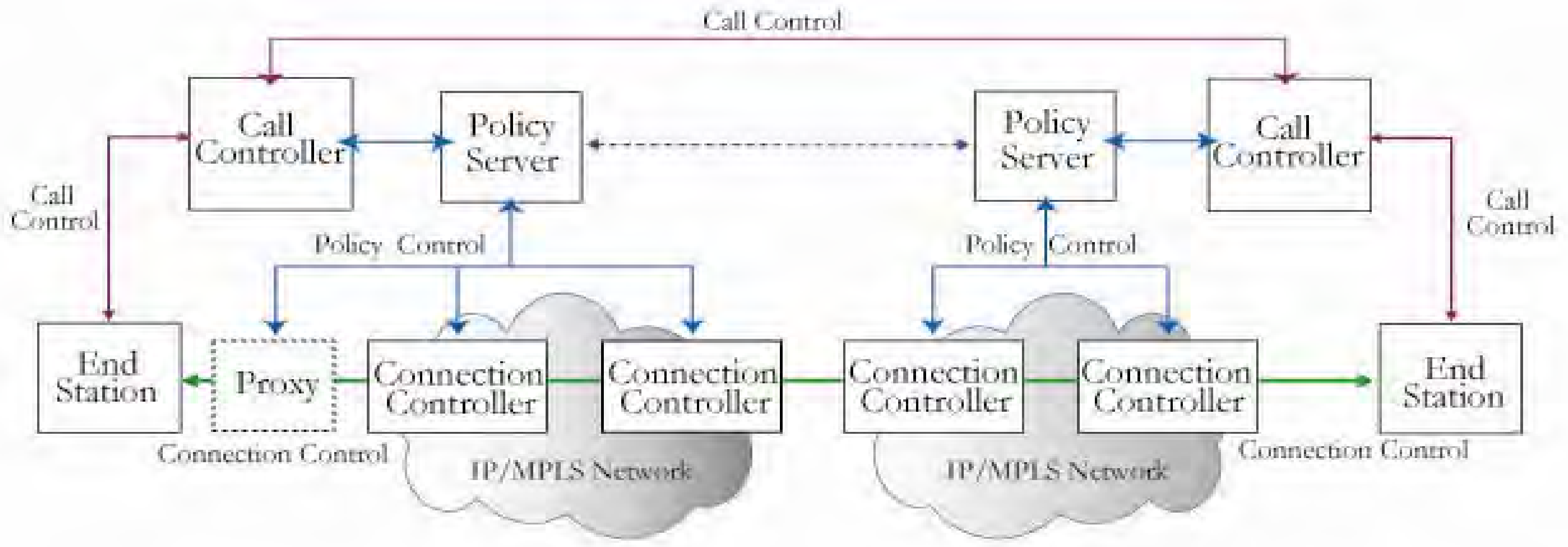
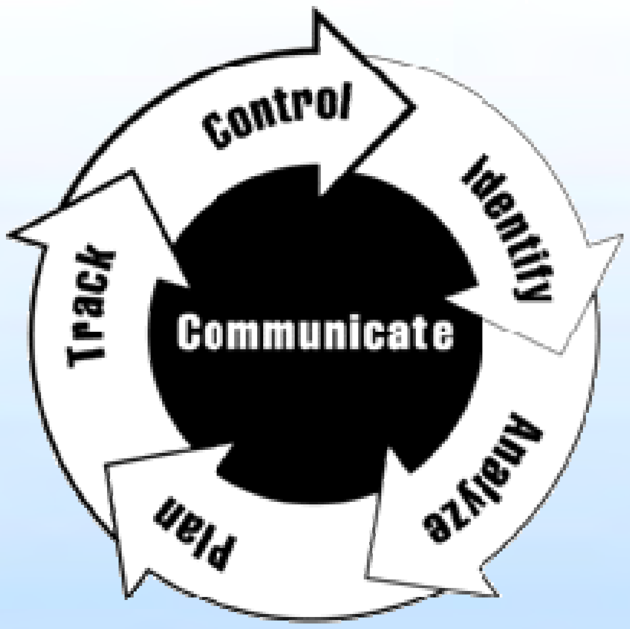


Figure 1. Reference QoS Control Architecture

Project Methodology

- New Distributed Network Monitoring Strategy
  - Comprehensive Network Usage Measurement (IETF IPFIX WG)
  - User Data Loss Detection (IETF RFC 2680)
  - Generic One-way Delay Monitor (IETF RFC 2679)
  - User Data Jitter Measurements (IETF RFC 3393)
  - Comprehensive Reachability Monitor (IETF RFC 2678)
  - Capacity/Utilization Monitor (IETF RFC 3148)
  - High Performance (OC-192) IPv4/IPv6 Passive Approach
- Establish Comprehensive Audit (IETF RTFM, ITU TMN)
- Utilize Uniform Data Collection (IETF IPFIX, ITU TMN)
- Perform fundamentally sound statistical analysis
- To Enable Effective Network Optimization

NTAIS FDO Optimization



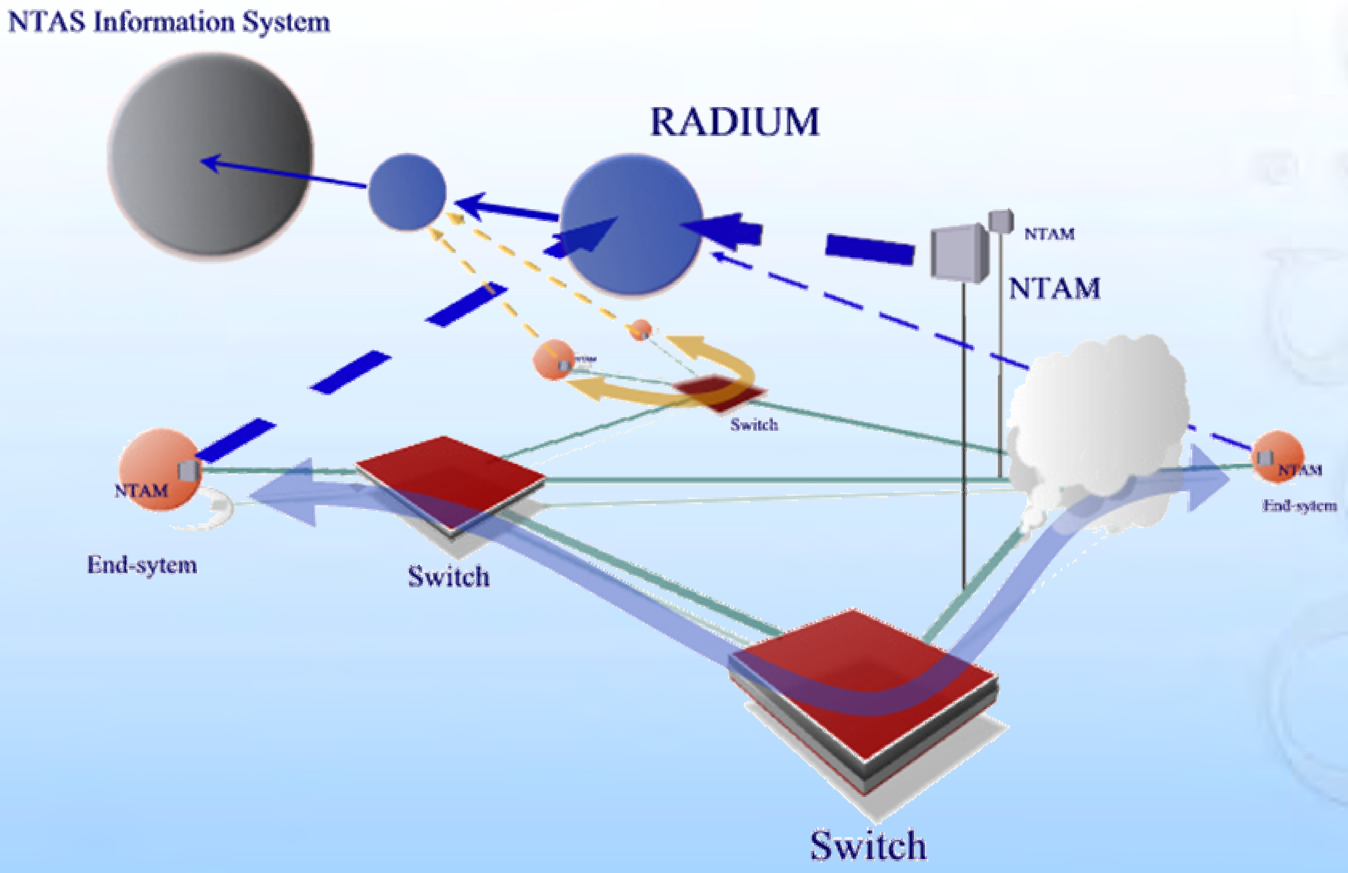
Function	**Description**	
Identify	Discover and Identify comprehensive network behavior.	Collect and Process Network Behavioral Data
Analyze	Collect and transform data into optimization metrics. Establish baselines, occurrence probabilities, and prioritize efforts.	
Plan	Establish optimization criteria (both present and future) and implement actions, if needed. This could involve reallocation of network resources, physical modifications, etc.	Provide information and feedback internal and external to the project on the optimization outcomes as events.
Track	Monitor network behavioral indicators to realize an effect.	
Control	Correct for deviations from the criteria.	

10 October 2005

Gargoyle Probe

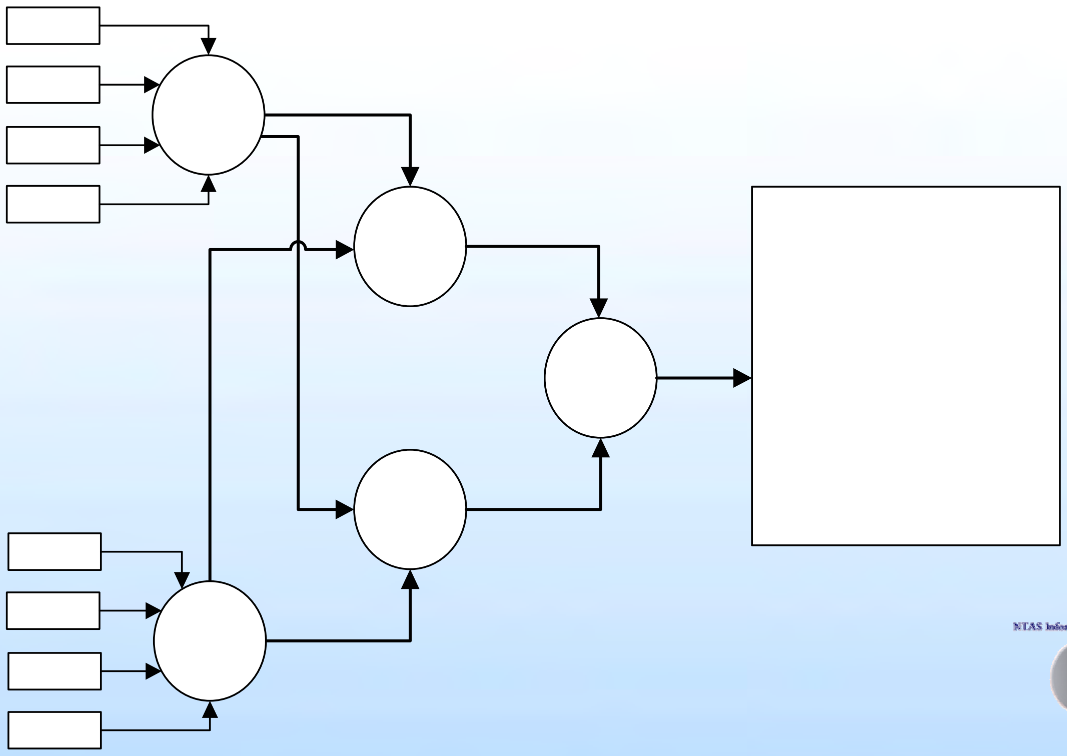
- Comprehensive Passive Real-Time Flow Monitor
  - User Plane and Control Plane Transaction Monitoring
  - Reporting on System/Network QoS status with every use
    - Capacity, Reachability, Responsiveness, Loss, Jitter
    - ICMP, ECN, Source Quench, DS Byte, TTL
- Multiple Flow Strategies
  - Layer 2, MPLS, VLAN, IPv4, IPv6, Layer 4 (TCP, IGMP, RTP)
- Small Footprint
  - 200K binary
- Performance
  - OC-192, 10GB Ethernet, OC-48, OC-12, 100/10 MB Ethernet, SLIP
  - POS, ATM, Ethernet, FDDI, SLIP, PPP
  - > 1.2 Mpkts/sec Dual 2GHz G5 MacOS X.
  - > 800Kpkts/sec Dual 2GHz Xeon Linux RH Enterprise
- Supporting Multiple OS's
  - Linux, Unix, Solaris, IRIX, MacOS X, Windows XP

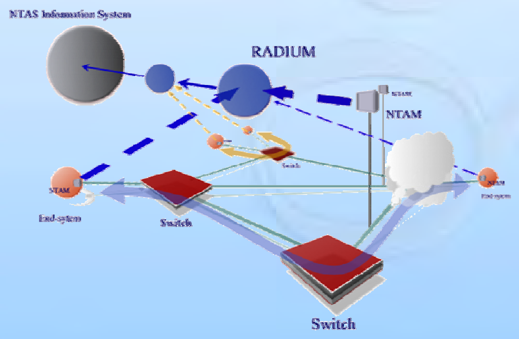
NTAS Architecture



10 October 2005

NTAS Distributed Architecture



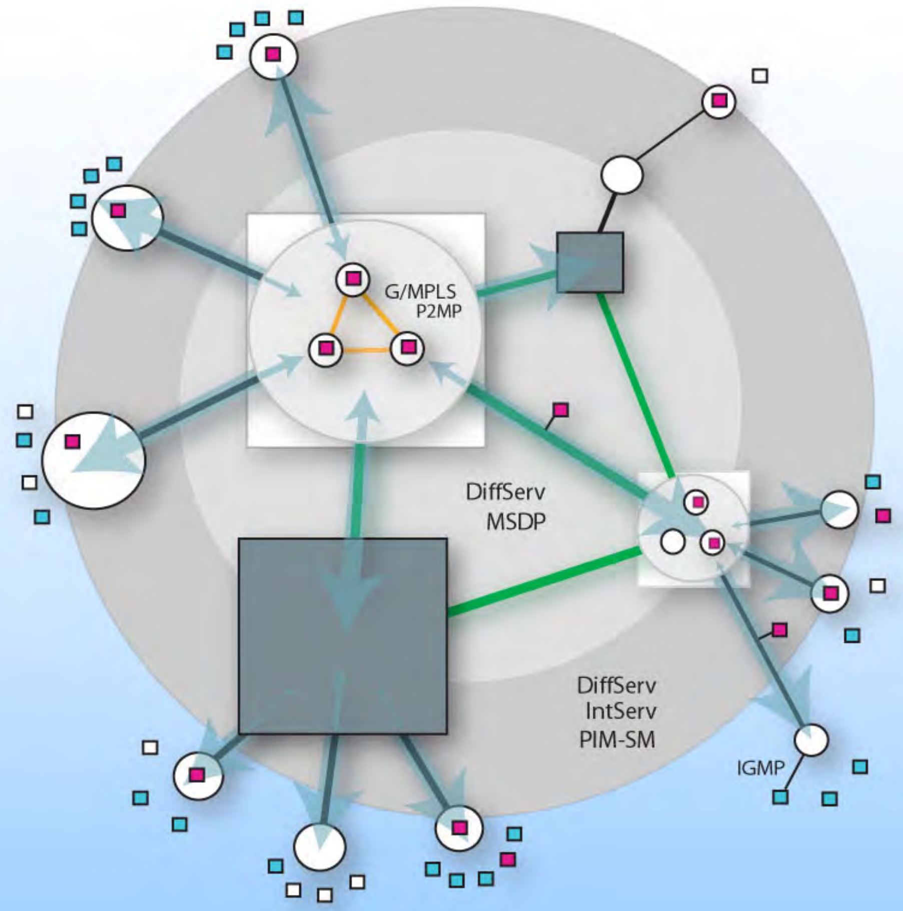


10 October 2005

NTAM

Unicast/Multicast QoS Monitor Strategies

Mixed Black-box White-box Approach



PIM-SM - IntServ	▬
MSDP - DiffServ	▬
P2MP - G/MPLS	▬
Core Transport	○
Multicast Router	○
NTAIS Gargoyle	■
Multicast Sender/Receiver	□
Multicast Traffic	➤

So, …, what is a flow?

- Classic 5-Tuple IP flow
- Encrypted VPN IP-Sec Tunnel
- MPLS based Label Switched Path (LSP)
- ATM Virtual Circuit
- PPP Association
- Routing Protocol Peer Adjacency
- Multicast Group Join Request/Reply
- Abstract Object <-> Abstract Object

And what metrics?

- Rate, Load, Bytes, Pkts, Goodput, Max Capacity
- Unidirectional? Bidirectional?
  - Connectivity, Reachability
  - RTT, One-way Delay
- Loss, Packet Size, Jitter, Retransmission Rate
- Protocol specific values (flags, sequence #)
- DS Code points
- TTL, Flow IDs
- Routing Flap Metrics
- Hello Arrival Rates

How Should They Be Transported

- Push/Pull?
- Reliable/Unreliable
- Unicast/Multicast
- Stream/Block/Datagram?
- Encrypted? Authenticated?

Argus

- Argus started 1990 – Georgia Tech
- Redesigned CERT/SEI/CMU – 1993
- Version 1.0 Open Source – 1995
  - Over 1M downloads
    - ~100,000 estimated sites worldwide
    - Unknown sites in production
- Supports 13 Type P and P1/P2 Flows
  - http://qosient.com/argus/flow.htm
- 117 Element Attribute Definitions
  - http://qosient.com/argus/Xml/ArgusRecord_xsd/ArgusRecord.htm

Argus Transport

- Pure Pull Strategy
  - Simplifies Probe Design
- Reliable Stream Transport (TCP)
  - Can support UDP/Multicast Datagram
- Supports TLS “On the Wire” Strong Authentication/Confidentiality
  - Probe Specifies Security Policy

Maybe Incompatible with IPFIX

- Template strategy can’t work with all the combinations of flow types supported.
- Distribution strategies make it even harder.
- Lack of identifiers to support flow objects
- Missing metric types.
- Vendor specific support is minimal
- Resulting in no motiviation to adopt.

IP Flow Information eXport (IPFIX)

elisa.boschi@hitachi-eu.com

{boschi, zseby, mark, hirsch}@fokus.fraunhofer.de

Outline

- IPFIX
- Terminology
- Applicability
- Initial Goals
- Current Status
  - *Rough consensus (Internet-Drafts and RFCs)*
  - *Running code (Implementations)*
- Conclusions

IP Flow Information eXport

- General data transport protocol
- Flexible flow key (selection)
- Flexible flow export - TEMPLATE BASED
  - New fields can be added to flow records without changing the structure of the record format
  - The collecor can always interpret flow records
  - external data format description → compact encoding
- Efficient data representation
  - Extensible (future attributes to be added)
  - Flexible (customisable)
  - Independent (of the Transport protocol)

HITACHI Inspire the Next FOKUS

Terminology

- A TEMPLATE is an ordered sequence of <type,length> pairs
  – specify the structure and semantics of a particular set of information (**Information Elements**)

- DATA RECORDS contain values of parameters specified in a template record

- OPTION RECORDS define the
  – structure and interpretation of a data record
  – how to scope the applicability

The protocol

- Unidirectional (push mode)
- The exporter sends data (and option) templates
  - Information Elements descriptions
- Information Elements are sent in network byte order

Applicability

- Target applications requiring flow-based IP traffic measurements (RFC 3917)
  - Usage-based accounting
  - Traffic profiling
  - Attack/intrusion detection
  - QoS monitoring
  - Traffic engineering

- Other applications (AS):
  - Network planning
  - Peering agreements

Attack / intrusion detection

- IPFIX provides input to attack / intrusion detection functions:
  - Unusually high loads
  - Number of flows
  - Number of packets of a specific type
  - Flow volume
  - Source and destination address
  - Start time of flows
  - TCP flags
  - Application ports

Initial Goals 1/4

- *Define the notion of a "standard IP flow"*

A Flow is a set of IP packets passing an Observation Point in the network during a certain time interval. All packets belonging to a particular flow have a set of common properties defined as the result of applying a function to the values of:

- *One or more packet header field (e.g. dest. IP address), transport header field (e.g. dest. port number), or application header field (e.g. RTP header fields RTP-HDRF)*
- *One or more characteristics of the packet itself (e.g. # of MPLS labels)*
- *One or more fields derived from packet treatment (e.g. next hop IP address)*

Initial Goals 2/4

- *Devise data encodings that support analysis of IPv4 and IPv6 unicast and multicast flows…*
  - IPFIX Information Model
    - formal description of IPFIX information elements (fields), their name, type and additional semantic information

- *Consider the notion of IP flow information export based upon packet sampling*
  - The **flow definition** includes packets selected by a sampling mechanism
  - Through **option templates**, the configuration sampling parameters can be reported

Initial Goals 3/4

- *Identify and address any security concerns affecting flow data.*
  - Disclosure of flow info data
  - Confidentiality → IPSec and TLS
  - Forgery of flow records
  - Authentication and integrity → IPSec and TLS
- *Specify the transport mapping for carrying IP flow information* → SCTP / SCTP-PR
  - Reliable (or partially reliable)
  - Congestion aware
  - Simpler state machine than TCP

HITACHI Inspire the Next FOKUS

Initial Goals 4/4

- *Ensure that the flow export system is reliable (minimize the likelihood of flow data being lost and to accurately report such loss if it occurs).*
  - – SCTP, TCP
  - – UDP
    - Templates are resent at a regular time interval
  - – Sequence numbers

Current status

- Internet-Drafts (~ sent to the IESG):
  - Architecture for IP Flow Information Export
  - Information Model for IP Flow Information Export
  - IPFIX Protocol Specification
  - IPFIX Applicability
- Request For Comments:
  - Requirements for IP Flow Information Export (RFC 3917)
  - Evaluation of Candidate Protocols for IP Flow Information Export (IPFIX) (RFC 3955)

Other related drafts

- Export of per packet information with IPFIX
  - E.Boschi, L.Mark draft-boschi-export-perpktinfo-00.txt
- IPFIX aggregation
  - F.Dressler, C.Sommer, G.Munz draft-dressler-ipfix-aggregation-01.txt
- Simple IPFIX Files for Persistent Storage
  - B.Trammell draft-trammell-ipfix-file-00.txt
- IPFIX templates for common ISP usage
  - E.Stephan, E. Moureau draft-stephan-isp-templates-00.txt
- IPFIX Protocol Specifications for Billing
  - B.Claise, P.Aitken, R.Stewart draft-bclaise-ipfix-reliability-00.txt
- IPFIX Implementation Guidelines

HITACHI Inspire the Next FOKUS

„Running code“

- At least 6 different IPFIX implementations
  - Ours is open source: http://www.6qm.org/downloads.php
- Implementers mailing list
- Interoperability events
  - July 2005, Paris (http://www.ist-mome.org)
  - Further tests planned
- Implementation guidelines in preparation

Conclusions

- IPFIX is the upcoming standard for (IP) flow information export
- Allows common analysis tools
- Data exchange

... questions?

IPFIX message format

- IPFIX message
  - message header
  - 1 or more {template, option template, data} sets

- A TEMPLATE is an ordered sequence of <type, length> pairs used to completely specify the structure and semantics of a particular set of information
  - (unique by means of a template ID)
  - DATA RECORDS contain values of parameters specified in a template record
  - Field values are encoded according to their data type specified in IPFIX-INFO
  - OPTION RECORDS define the structure and interpretation of a data record including how to scope the applicability

INFORMATION ELEMENTS

- INFORMATION ELEMENTS are descriptions of attributes which may appear in an IPFIX record
  - IANA assigned
  - Defined in the Information Model
  - Enterprise specific (proprietary I.E.)
- Variable Length I.E.
  - The length is carried in the information element content itself
- The type associated with an IE
  - indicates constraints on what it may contains
  - determines the valid encoding mechanisms for use in IPFIX
- I.E.s must be sent in network byte order (big endian)

INFORMATION ELEMENTS

- The elements are grouped into 9 groups according to their semantics and their applicability:

1. Identifiers
2. Metering and Exporting Process Properties
3. IP Header Fields
4. Transport Header Fields
5. Sub-IP Header Fields
6. Derived Packet Properties
7. Min/Max Flow Properties
8. Flow Time Stamps
9. Per-Flow Counters
10. Miscellaneous Flow Properties

$\Big\}$ (3.–6.) can serve as Flow Keys

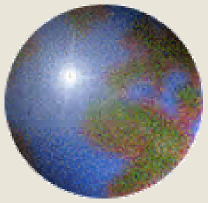
(used for mapping packets to Flows)

HITACHI Inspire the Next FOKUS

Requirements for the data model

- IPFIX is intended to be deployed in high speed routers and to be used for exporting at high flow rates
- → Efficiency of data representation
- How data is represented = data model
- EXTENSIBLE
  - For future attributes to be added
- FLEXIBLE
  - Concerning the attributes (customisable)
- INDEPENDENT
  - Of the transport protocol

HITACHI Inspire the Next FOKUS



Data Mining NetFlow

So What's Next?

Mark E Kane

FloCon 2005

20 September 05

Objectives

- Data Mining, very briefly
- Frequency Patterns
- Discoveries
- Realizations
- Changes Made

Data Mining

Data Mining – automated extraction of previously unknown data that is interesting and potentially useful.

Cost of Participating in Data Mining

Reality	Result of Data Mining	Example Analyst Hours	Example Investigator Hours	Example SysAdmin Hours	Result
YES	YES	10	10	10	Crime Prevented / Prosecuted
NO	NO	0	0	0	-
YES	NO	∞	∞	∞	Time Lost to Investigate and Clean Up After Crime
NO	YES	10	10	10	Red Haring

Complexity of Mining NetFlow

- Shear Volume
- Complex Protocol Analysis
- Ambiguous Interpretations
- Very Smart Adversaries

Common Investigator Issues

- Undermanned and overworked
- Varied knowledge base
- Does not own networks
- No direct reporting structure

Data Mining Techniques

Primary Techniques

- Rule and Tree Induction
- Characterization
- **Classification**
- Regression
- Association
- **Clustering**

Other Techniques

- Dependency Modeling
- **Change Detection**
- Trend Analysis
- Deviation Detection
- **Link Analysis**
- Pattern Analysis
- Spatiotemporal Data Mining
- Mining Path Traversal Patterns
- **Mining Sequential/Frequent Patterns**

Uncertain Reasoning Techniques

- Fuzzy Logic
- Neural Networks
- Bayesian Networks
- Genetic Algorithms
- Rough Set Theory

Frequency Patterns

Mining Frequent Patterns in Data Streams in Multiple Time Granularities (Giennella, Han, Pei, Yan, and Yu)

- Support Decision Making
- Past Less Significant than Present
- Record Reduction
- Time Tilted Windows

Interpreting Time-Tilted Windows

	DAY								
Window			0		1		2		3
Transition		N	Y	N	Y	N	Y	N	Y
Size		1	1	2	2	4	4	8	8
	Monday	9							
	Tuesday	15	9						
	Wednesday	6		12					
	Thursday	6	6	12					
	Friday	12		6	12				
	Saturday	16	12	6	12				
	Sunday	6		14		9			
	Monday	12	6	14		9			
	Tuesday	15		9	14	9			

Day 1: 9 events

Day 2: 15 events (two buckets)

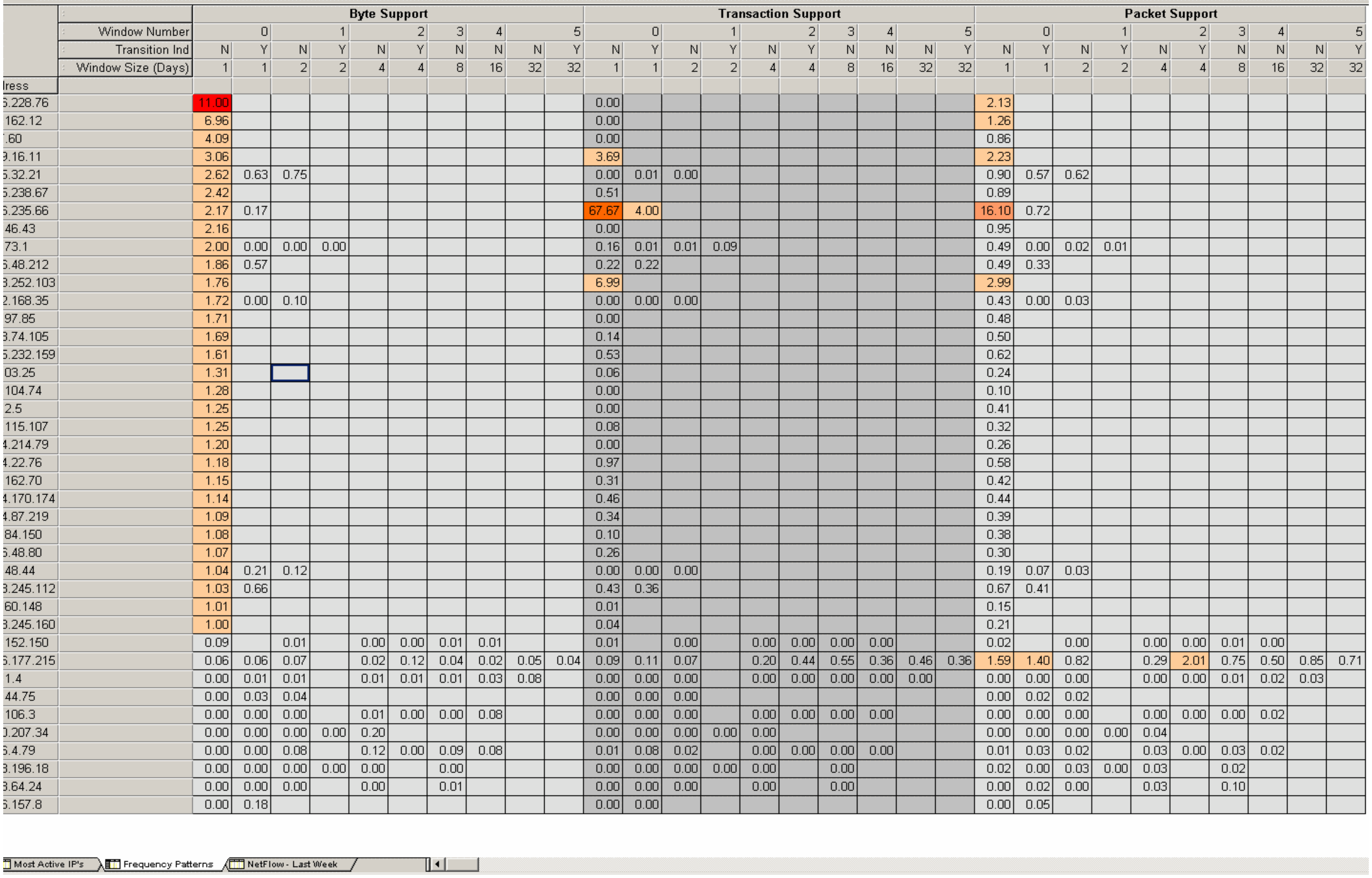
Day 3: 6 events (two buckets)

Day 4: 6 events (two buckets)

Day 5: 16 events (three buckets)

Day 6: 12 events (four buckets)

Presenting Frequency Patterns



		Byte Support										Transaction Support										Packet Support									
	Window Number	0		1		2		3	4	5		0		1		2		3	4	5		0		1		2		3	4	5	
	Transition Ind	N	Y	N	Y	N	Y	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	Y	N	Y	N	Y	N	Y	N	N	N	Y
	Window Size (Days)	1	1	2	2	4	4	8	16	32	32	1	1	2	2	4	4	8	16	32	32	1	1	2	2	4	4	8	16	32	32
ress																															
5.228.76		11.00										0.00										2.13									
162.12		6.96										0.00										1.26									
.60		4.09										0.00										0.86									
3.16.11		3.06										3.69										2.23									
5.32.21		2.62	0.63	0.75								0.00	0.01	0.00								0.90	0.57	0.62							
5.238.67		2.42										0.51										0.89									
5.235.66		2.17	0.17									67.67	4.00									16.10	0.72								
46.43		2.16										0.00										0.95									
73.1		2.00	0.00	0.00	0.00							0.16	0.01	0.01	0.09							0.49	0.00	0.02	0.01						
5.48.212		1.86	0.57									0.22	0.22									0.49	0.33								
3.252.103		1.76										6.99										2.99									
2.168.35		1.72	0.00	0.10								0.00	0.00	0.00								0.43	0.00	0.03							
97.85		1.71										0.00										0.48									
3.74.105		1.69										0.14										0.50									
5.232.159		1.61										0.53										0.62									
03.25		1.31										0.06										0.24									
104.74		1.28										0.00										0.10									
2.5		1.25										0.00										0.41									
115.107		1.25										0.08										0.32									
4.214.79		1.20										0.00										0.26									
4.22.76		1.18										0.97										0.58									
162.70		1.15										0.31										0.42									
4.170.174		1.14										0.46										0.44									
4.87.219		1.09										0.34										0.39									
84.150		1.08										0.10										0.38									
5.48.80		1.07										0.26										0.30									
48.44		1.04	0.21	0.12								0.00	0.00	0.00								0.19	0.07	0.03							
3.245.112		1.03	0.66									0.43	0.36									0.67	0.41								
60.148		1.01										0.01										0.15									
3.245.160		1.00										0.04										0.21									
152.150		0.09		0.01		0.00	0.00	0.01	0.01			0.01		0.00		0.00	0.00	0.00	0.00			0.02		0.00		0.00	0.00	0.01	0.00		
5.177.215		0.06	0.06	0.07		0.02	0.12	0.04	0.02	0.05	0.04	0.09	0.11	0.07		0.20	0.44	0.55	0.36	0.46	0.36	1.59	1.40	0.82		0.29	2.01	0.75	0.50	0.85	0.71
1.4		0.00	0.01	0.01		0.01	0.01	0.01	0.03	0.08		0.00	0.00	0.00		0.00	0.00	0.00	0.00	0.00		0.00	0.00	0.00		0.00	0.00	0.01	0.02	0.03	
44.75		0.00	0.03	0.04								0.00	0.00	0.00								0.00	0.02	0.02							
106.3		0.00	0.00	0.00		0.01	0.00	0.00	0.08			0.00	0.00	0.00		0.00	0.00	0.00	0.00			0.00	0.00	0.00		0.00	0.00	0.00	0.02		
0.207.34		0.00	0.00	0.00	0.00	0.20						0.00	0.00	0.00	0.00	0.00						0.00	0.00	0.00	0.00	0.04					
5.4.79		0.00	0.00	0.08		0.12	0.00	0.09	0.08			0.01	0.08	0.02		0.00	0.00	0.00	0.00			0.01	0.03	0.02		0.03	0.00	0.03	0.02		
3.196.18		0.00	0.00	0.00	0.00	0.00		0.00				0.00	0.00	0.00	0.00	0.00		0.00				0.02	0.00	0.03	0.00	0.03		0.02			
3.64.24		0.00	0.00	0.00		0.00		0.01				0.00	0.00	0.00		0.00		0.00				0.00	0.02	0.00		0.03		0.10			
5.157.8		0.00	0.18									0.00	0.00									0.00	0.05								

Data Mining Discoveries

- Failed email servers
- Previously, unknown trusted relationships
- Encryption without authentication
- Possible, but unproven intrusions

Data Mining Results

Frustrated Investigators

Frustrated Analysts

One Very Frustrated Developer

Changes to Employ Data Mining

Establish common basis of understanding

Establish criteria for reporting

- Geo-Resolution
- Timeliness
- Volume

Establish reporting procedures

Questions

Mark Kane

mkane @ ddktechgroup.com

Detecting Distributed Attacks using Network-Wide Flow Traffic

Anukool Lakhina
Dept. of Computer Science,
Boston University
anukool@cs.bu.edu

Mark Crovella
Dept. of Computer Science,
Boston University
crovella@cs.bu.edu

Christophe Diot
Intel Research
Cambridge, UK
christophe.diot@intel.com

1. INTRODUCTION

Distributed denial of service attacks have become both prevalent and sophisticated. Botnet-driven attacks can be launched from thousands of worm-infected and compromised machines with relative ease and impunity today. The damage caused by such attacks is considerable: the 2004 CSI/FBI computer crime and security survey found that DDOS attacks are the second largest contributor to all financial losses due to cybercrime [3]. Further, distributed attacks are expected to increase both in sophistication and damage [1]. Containing distributed attacks is therefore a crucial problem, one that has not been adequately addressed.

One reason why distributed attacks are difficult to contain is because defenses against these attacks are typically deployed at edge networks, near the victim. Deploying defenses at the edge makes detecting attacks easier, since one simply needs to monitor incoming traffic volume for an unusually large burst. However, containing and mitigating such attacks from the edge is ineffective for two reasons. First, filtering the malicious attack traffic requires identifying the (potentially thousands of) attackers, which is complicated, especially if the source addresses are spoofed. Second, even if accurate filtering was feasbile at the edge, it cannot prevent attackers from consuming the victim's bandwidth, and denying service to legitimate users. Thus edge-based defenses against distributed attacks have limited value.

On the other hand, defending against distributed attacks at the backbone (*i.e.*, carrier networks) overcomes the hurdles of edge-based defenses. In principle, backbone networks can detect and identify the origins of malicious sources involved in a distributed attack that traverses the backbone. Thus backbone networks are well-suited to mitigate distributed attacks, before they cause harm to the victim at the edge. However, distributed attacks are challenging to detect in the backbone because they do not cause a visible, easily detectable change in traffic volume on individual backbone links. To effectively detect distributed attacks in the backbone, one therefore needs to simultaneously analyze all traffic across the network.

In this work, we present our methods to detect distributed attacks in backbone networks using sampled flow traffic data. Distributed attacks are traditionally viewed to be fundamentally more difficult to detect than single-source attacks. In contrast, we demonstrate that the more distributed an attack is,the better our methods are at detecting it. This is because our methods analyze correlations across all *network-wide* traffic simultaneously, instead of inspecting traffic on individual links in isolation. In addition, our methods are highly sensitive to the attack intensity; we show that attacks rates of less than 1% of the underlying traffic can be detected successfully by our methods.

The rest of this paper is organized as follows. In the next section we show how network-wide traffic summaries can be assembled, and present the data we have processed from the Abilene Internet2 backbone network. Then, in Section 3, we describe the multiway subspace method for detecting attacks in network-wide flow data. We evaluate our methods on actual DDOS attack traces in a series of experiments and present results in Section 4. Finally, we conclude in Section 5.

2. NETWORK-WIDE FLOW DATA

Our methods analyze all traffic that traverses the network. To obtain such network-wide flow traffic, we must collect the ensemble of origin-destination flows (OD flows) from a network. The traffic in an Origin-Destination flow is the set of traffic that enters the network at specific point (the origin) and exits the network at the destination. For this study, we assembled the set of OD flows for the Abilene network.

Abilene is the Internet2 backbone network, connecting over 200 US universities and peering with research networks in Europe and Asia. It consists of 11 Points of Presence (PoPs), spanning the continental US. We collected three weeks of sampled IP-level traffic flow data from every PoP in Abilene for the duration of December 8, 2003 to December 28, 2003. Sampling is periodic, at a rate of 1 out of 100 packets, and flow statistics are reported every 5 minutes; this allows us to construct traffic timeseries with bins of size 5 minutes.

To aggregate the IP flow data at the OD flow level, we must resolve the egress PoP for each flow record measured at a given ingress PoP. This egress PoP resolution is accomplished by using BGP and ISIS routing tables, as detailed in [2]. After this procedure is completed, there are 121 OD flows in Abilene.

Our final post-processing step constructs timeseries at 5 minute bins for traffic summaries of each OD flow. The traffic summary we use is the sample entropy of the four main traffic features (source IP, destination IP, source port and destination port). Sample entropy captures the *distribution* of each traffic feature in a manner that reveals unusual changes in the distribution. An analysis of the merits of distributional-based analysis of traffic features for anomaly diagnosis can be found in [6].

To summarize, the network-wide flow traffic we study is the multivariate timeseries of sample entropy of traffic features for the ensemble of Abilene's OD flows.

3. THE MULTIWAY SUBSPACE METHOD

To detect distributed attacks, it is necessary to examine network-wide traffic - as captured by the set of OD flows - simultaneously. The multiway subspace method accomplishes this task and is described in [6]; we review the main ideas here.

Thinning Rate	0	10	100	1000	10000
Attack Intensity (pps)	2.75e4	2.75e3	275	27.5	25.9
Attack Intensity (%)	93%	57%	12%	1.3%	0.13%

Table 1: Intensity of injected attack, in # pkts/sec (pps) and percent of (single) OD flow traffic.

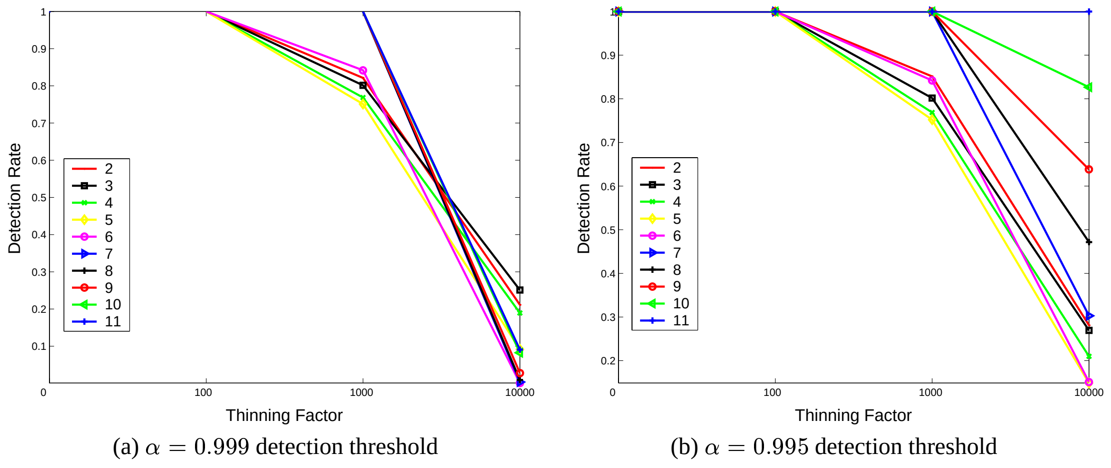


Figure 1: Detection results from injecting multi-OD flow attacks (across 2 to 11 OD flows).

The multiway subspace method separates the ensemble of OD flow timeseries into normal and anomalous attributes. Normal traffic behavior is determined directly from the data, as those temporal patterns that are most common to the set of OD flows. This extraction of common trends is achieved by Principal Component Analysis (PCA). As shown in [7], PCA can be used to decompose the set of OD flows into their constitutent common temporal patterns.

A key result of [7] was that a handful of dominant temporal patterns are common to the hundreds of OD flows. The multiway subspace method exploits this result by designating these domiant trends as normal, and the remaining temporal patterns as anomalous. As a result, each OD flow can be reconstructed as a sum of normal and anomalous components. In particular, we can write, $\mathbf{x} = \hat{\mathbf{x}} + \tilde{\mathbf{x}}$, where $\mathbf{x}$ denotes the traffic of all the OD flows at a specific point in time, $\hat{\mathbf{x}}$ is the reconstruction of $\mathbf{x}$ using only the dominant temporal patterns, and $\tilde{\mathbf{x}}$ contains the residual traffic.

Once this separation is completed, detection of unusual events requires monitoring the size (ℓ_2 norm) of $\tilde{\mathbf{x}}$. The size of $\tilde{\mathbf{x}}$ measures the degree to which $\mathbf{x}$ is anomalous. Statistical tests can then be formulated to test for unusual large $\|\tilde{\mathbf{x}}\|$, based on a desired false alarm rate [5].

As demonstrated in [6], the multiway subspace method can detect a broad spectrum of anomalous events, at a low false alarm rate. Further, these anomalies can span multiple traffic features, and also occur in multiple OD flows. Our focus in this work is to specifically evaluate the power of network-wide traffic analysis, via the multiway subspace method, to detect distributed attacks.

4. DETECTION RESULTS

In this section we specifically study how effective our methods are at detecting distributed attacks. We first describe our experimental setup, where we inject traces of a known distributed denial of service attack in the Abilene network-wide flow data. Next, we present results from applying the multiway subspace method to detect these injected attacks.

4.1 Injecting Distributed Attacks

To evaluate our detection method, we decided to use an actual distributed denial of service attack packet trace and superimpose it onto our Abilene flow data in a manner that is as realistic as possible. This involved a number of steps which we describe below.

We use the distributed denial of service attack trace collected and described in [4]. This 5-minute trace consists of packet headers without any sampling. It was collected at a Los Nettos regional ISP in 2003, and so exemplifies an attack on an edge network. We extracted the attack traffic from this attack trace by identifying all packets directed to the victim. We then mapped header fields in the extracted packets to appropriate values for the Abilene network.

Then, to construct representative distributed attacks, we divided the attack trace into k smaller traces, based on uniquely mapping the set of source IPs in the attack trace onto k different origin PoPs of Abilene. The splitting was performed so that each of the k groups has roughly the same amount of traffic. Next, we injected this k-partitioned trace into k OD flows sharing the same destination PoP (the victim of the DDOS attack). For each destination PoP, we injected the k OD flow attack into all possible combinations of k sources, *i.e.*, $\binom{p}{k}$ combinations where $p = 11$ is the number of PoPs in the Abilene network. We repeated this set of experiments for every destination PoP in the network; thus for a given choice of $2 \leq k \leq p$, we performed $\binom{p}{k} \cdot p$ total experiments. For each multi-OD flow injection, we recorded if the multiway subspace method detected the attack.

Finally, we repeated the entire set of experiments at different thinning rates to measure the sensitivity of the detection methods to lower intensity DDOS attacks. We thinned the original attack trace by selecting 1 out of every N packets, then extracted the attack and injected it into the Abilene OD flows, as described above. The resulting attack intensity for the various thinning rates is shown in Table 1. The table also shows the percent of all packets in the resulting OD flow that was due to the injected anomaly.

While these multi-OD flow experiments are designed to span a number of origin PoPs sharing a common destination PoP, our detection methods do not assume any fixed topological arrangement on the malicious OD flows. The results from these experiments give us insight into the performance of the multiway subspace method in detecting attacks that are dwarfed in individual OD flows, but are only visible network-wide, across multiple OD flows.

4.2 Results

We now present results on using the multiway subspace method to detect multi-OD flow attacks. The detection rates (averaged over the entire set of experiments) from injecting DDOS attacks spanning $2 \leq k \leq 11$ OD flows are shown in Figure 1. Figure 1(a) and (b) present results when the detection threshold is α=0.999 (equivalent to asking for a false alarm rate of 1-α, or 1 in 1000) and α=0.995 (equivalent to a false alarm rate of 5 in 1000).

Both figures show that we can effectively detect attacks spanning a large number of OD flows. In fact, the detection rates are generally higher for larger k, *i.e.,* for attacks that span a larger number of OD flows. For example, in Figure 1(a) we detect 100% of DDOS attacks that are split evenly across all the 11 possible origins PoPs in the Abilene network, even at a thinning rate of 1000. From Table 1, the average intensity of the DDOS attack trace in each of the 11 OD flows at a thinning rate of 1000 is $\frac{27.5}{11} = 2.5$ packets/sec.

In Figure 1(b), we relaxed the detection threshold to $\alpha = 0.995$. In this setting, we detect about 82% of all DDOS attack traffic spanning 10 OD flows, at thinning rates of even 10000, which corresponds to an attack with intensity of 0.259 packets/sec in each of the 10 participating OD flows individually. Such low-rate attacks are a tiny component of any single OD flow, and so are only detectable when analyzing multiple OD flows simultaneously.

Thus, the results here underscore the power of network-wide analysis via the multiway subspace method.

5. CONCLUSIONS

Distributed attacks are an important problem facing networks today. We argue that distributed attacks are best mitigated at the backbone. Detecting distributed attacks at the backbone requires departing from traditional single-link traffic analysis and adopting a *network-wide* view to traffic monitoring. In this work, we applied the multiway subspace method on network-wide flow data to detect distributed attacks in the Abilene backbone network. Through a series of controlled experiments, we demonstrated that the multiway subspace method is well suited to detect massively distributed attacks, even those with low attack intensity.

6. ACKNOWLEDGEMENTS

We thank Alefiya Hussain for providing the multi-source denial of service attack trace.

This work was supported in part by NSF grants ANI-9986397 and CCR-0325701, and by Intel.

7. REFERENCES

[1] CERT Research. 2004 Annual Report. At `www.cert.org/archive/pdf/cert_rsrch_annual_rpt_2004.pdf`.

[2] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational IP networks: Methodology and experience. In *IEEE/ACM Transactions on Neworking*, pages 265–279, June 2001.

[3] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson. 2004 CSI/FBI Computer Crime and Security Survey. Available at `www.gocsi.com/forms/fbi/csi_fbi_survey.jhtml`, 2004.

[4] A. Hussain, J. Heidemann, and C. Papadopoulos. A Framework for Classifying Denial of Service Attacks. In *ACM SIGCOMM*, Karlsruhe, August 2003.

[5] A. Lakhina, M. Crovella, and C. Diot. Diagnosing Network-Wide Traffic Anomalies. In *ACM SIGCOMM*, Portland, August 2004.

[6] A. Lakhina, M. Crovella, and C. Diot. Mining Anomalies Using Traffic Feature Distributions. In *ACM SIGCOMM*, Philadelphia, August 2005.

[7] A. Lakhina, K. Papagiannaki, M. Crovella, C. Diot, E. D. Kolaczyk, and N. Taft. Structural Analysis of Network Traffic Flows. In *ACM SIGMETRICS*, New York, June 2004.

CANINE

A NetFlows Conversion/Anonymization Tool for Format Interoperability and Secure Sharing

Katherine Luo*, Yifan Li, Adam Slagell, William Yurick

SIFT Research Group

National Center for Supercomputing Applications (NCSA)

University of Illinois at Urbana-Champaign

FloCon05, Sep. 20, 2005

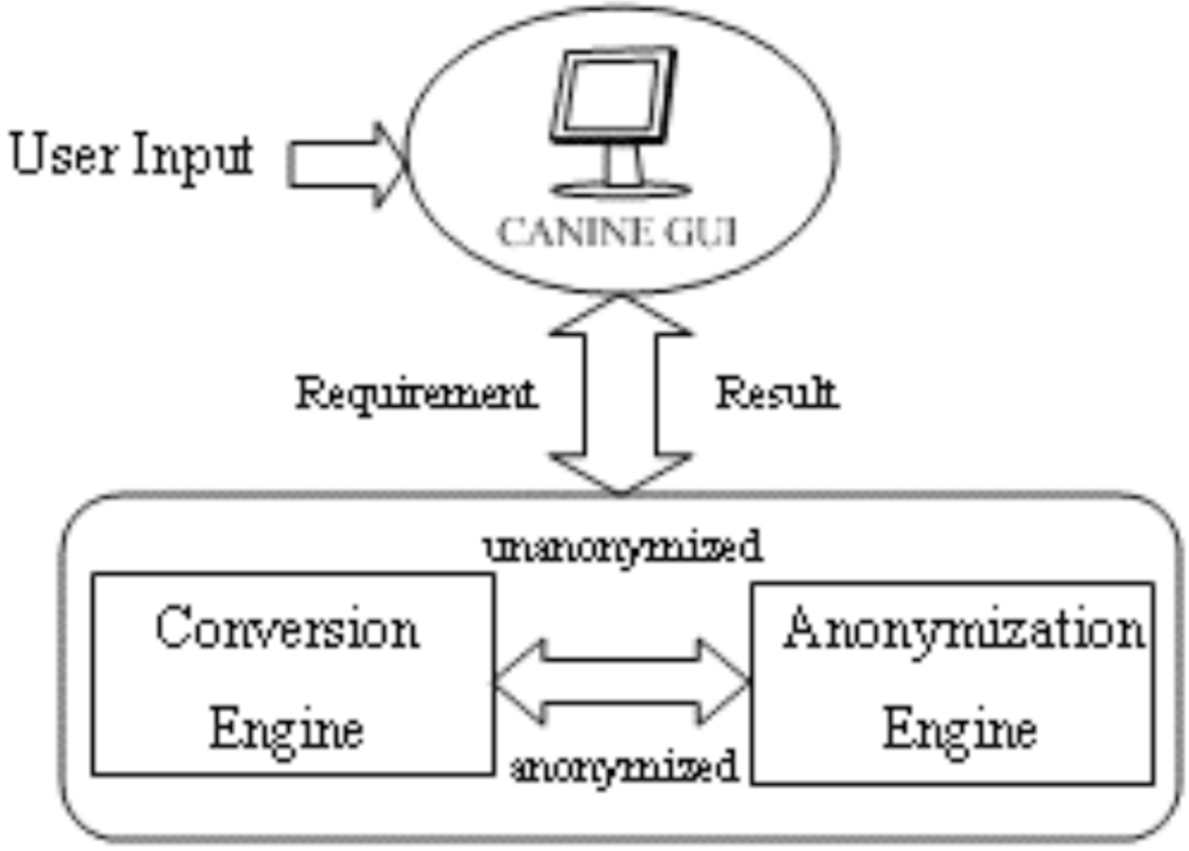
Motivations

- NetFlows in multiple, incompatible formats
  - Network security monitoring tools usually support one or two NetFlows format
  - Need conversion of NetFlows between different formats
- Sensitive network information hinders log sharing
  - Log sharing necessary for research and study
  - Need anonymization of sensitive data fields

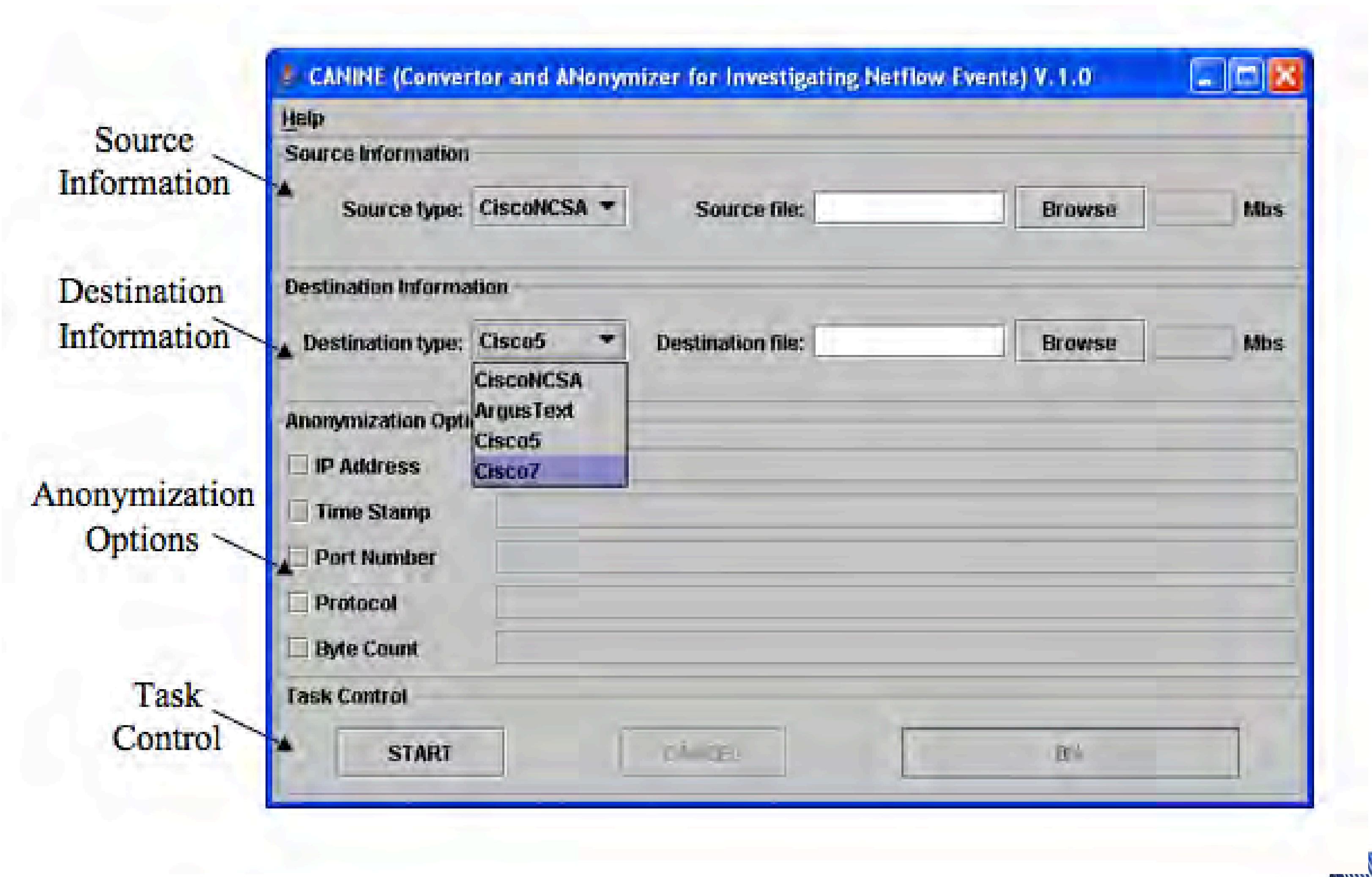
Our Solution: CANINE Tool

- CANINE: Converter and ANonymizer for Investigating Netflow Events
- Handles several NetFlow formats
  - Cisco V5 & V7, ArgusNCSA, CiscoNCSA, NFDump
- Anonymizes 5 types of data fields
  - IP, Timestamp, Port, Protocol and Byte Count
- Multiple anonymization levels
  - Various anonymization methods for some data field

System Architecture of CANINE



Main GUI of CANINE



CANINE (Convertor and ANonymizer for Investigating Netflow Events) V.1.0

Help

Source Information

Source type: CiscoNCSA | Source file: | Browse | Mbs

Destination Information

Destination type: Cisco5 | Destination file: | Browse | Mbs

CiscoNCSA
ArgusText
Cisco5
Cisco7

Anonymization Opti

- IP Address
- Time Stamp
- Port Number
- Protocol
- Byte Count

Task Control

START | CANCEL | 0%

Conversion & Anonymization Engine

- Conversion Engine
  - Parse the input NetFlow record into component data fields before anonymization
  - Reassemble the anonymized data component to desired NetFlow format

- Anonymization Engine
  - Contain a collection of anonymization algorithms
  - Anonymize data fields with designated methods

IP Address Anonymization

- Truncation
  - Zeroing out any number of LSBs
- Random Permutation
  - Generate a random IP number seeded by user input
- Prefix-preserving Pseudonymization
  - Match on n-bit prefix, based on Crypto-PAn

IP Address	Truncation (16-bit)	Random Permutation	Prefix-preserving
141.142.96.167	141.142.0.0	124.12.132.37	12.131.102.67
141.142.96.18	141.142.0.0	231.45.36.167	12.131.102.197
141.142.132.37	141.142.0.0	12.72.8.5	12.131.201.29

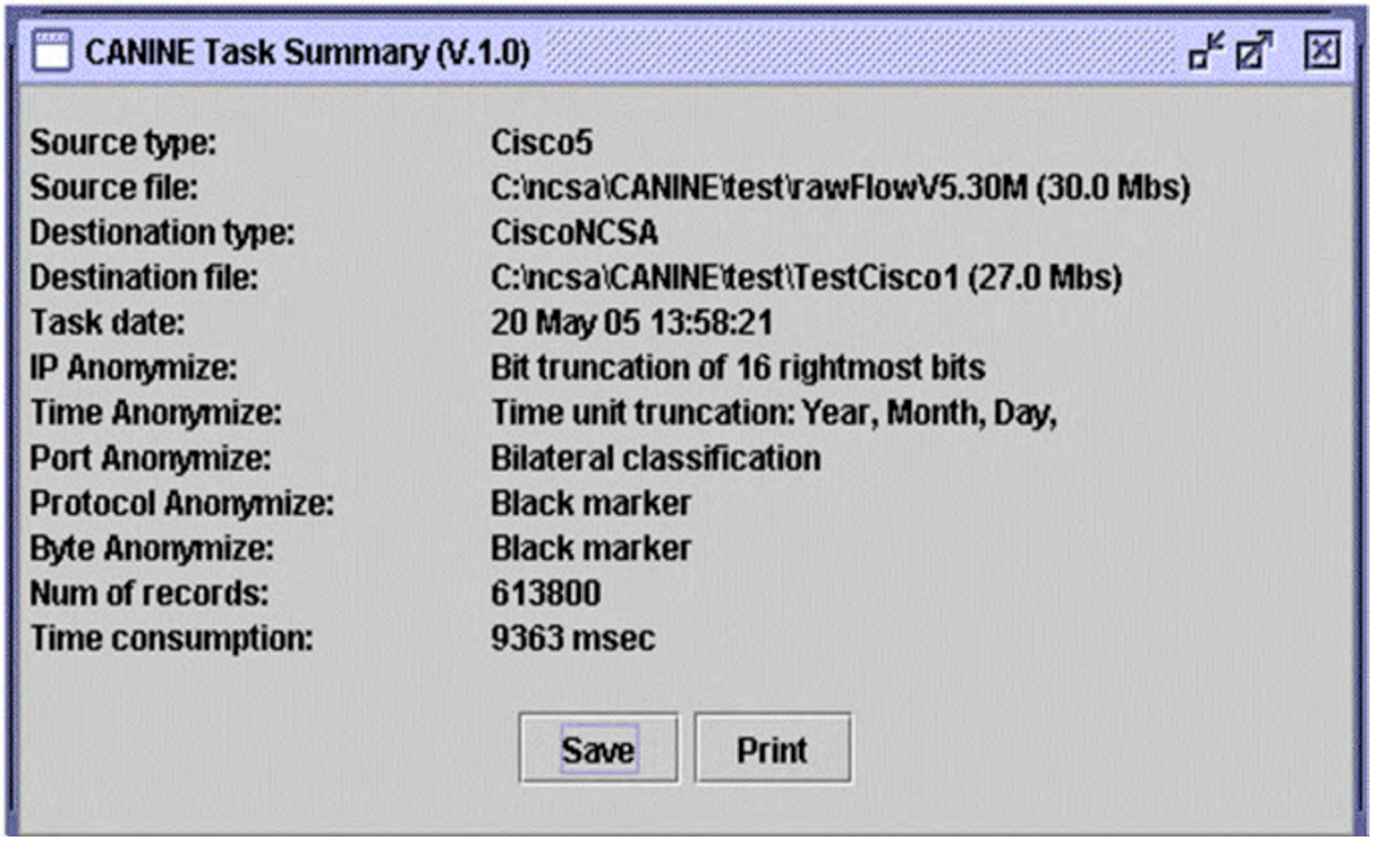
Timestamp Anonymization

- Time Unit Annihilation
  - Zeroing-out indicated subset of time units on end time
  - Start time is adjusted to keep the duration unchanged
- Random Time Shift
  - Pick a range for generating random shift
  - Shift all timestamps by the same amount
- Enumeration
  - Local sorting performs based on end time
  - Set the slide window size
  - Records sorted and equidistantly spaced

Port Number, Protocol, Byte Count Anonymization

- Port Number Anonymization
  - Bilateral classification
    - Replace with 0 or 65535 (the port smaller or larger than 1024)
  - Black marker
    - Replace with 0
- Protocol Anonymization
  - Black Maker
    - Replace with 255 (IANA reserved but unused number)
- Byte Count Anonymization
  - Black Marker
    - Replace with 0 (Impossible value in practice)

Task Summary Dialog



Source type:	Cisco5
Source file:	C:\ncsa\CANINE\test\rawFlowV5.30M (30.0 Mbs)
Destionation type:	CiscoNCSA
Destination file:	C:\ncsa\CANINE\test\TestCisco1 (27.0 Mbs)
Task date:	20 May 05 13:58:21
IP Anonymize:	Bit truncation of 16 rightmost bits
Time Anonymize:	Time unit truncation: Year, Month, Day,
Port Anonymize:	Bilateral classification
Protocol Anonymize:	Black marker
Byte Anonymize:	Black marker
Num of records:	613800
Time consumption:	9363 msec

Summary and Future Work

- CANINE addressed two problems
  - Convert and anonymize NetFlow logs
  - Unique due to multiple anonymization levels
- Modifications on CANINE
  - Config file alternative to GUI
  - Streaming mode processing
- Research on multiple levels of anonymization scheme
  - Utility of the anonymized log
  - Security of the anonymization schemes

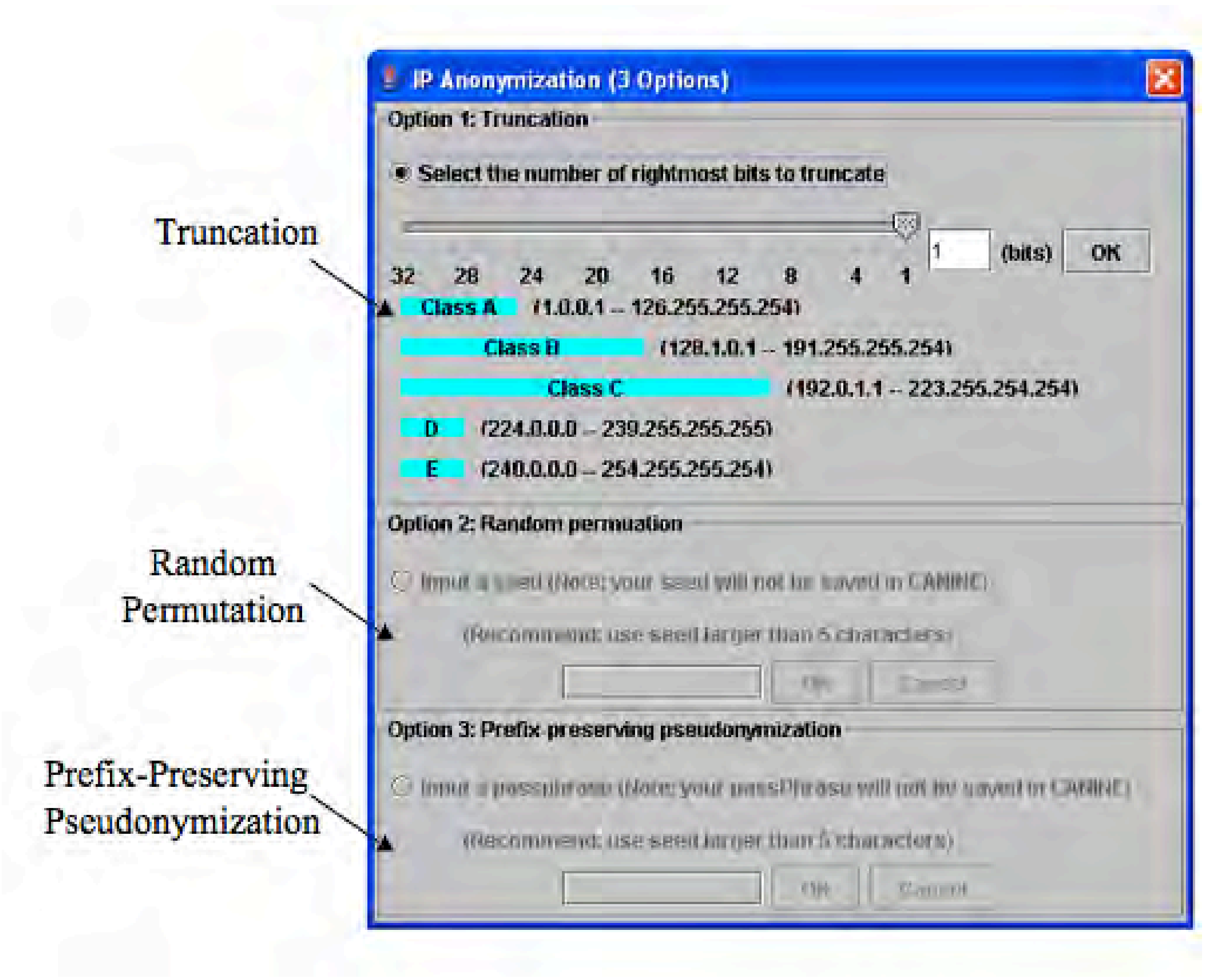
Download CANINE at

http://security.ncsa.uiuc.edu/distribution/CanineDownLoad.html

Thank you!

Questions?

IP Address Anonymization



IP Anonymization (3 Options)

Option 1: Truncation

[✓ Select the number of rightmost bits to truncate]

1 (bits) OK

32 28 24 20 16 12 8 4 1

Class A (1.0.0.1 -- 126.255.255.254)

Class B (128.1.0.1 -- 191.255.255.254)

Class C (192.0.1.1 -- 223.255.254.254)

D (224.0.0.0 -- 239.255.255.255)

E (240.0.0.0 -- 254.255.255.254)

Option 2: Random permuation

Input a seed (Note: your seed will not be saved in CANINE)

(Recommend: use seed larger than 6 characters)

Ok Cancel

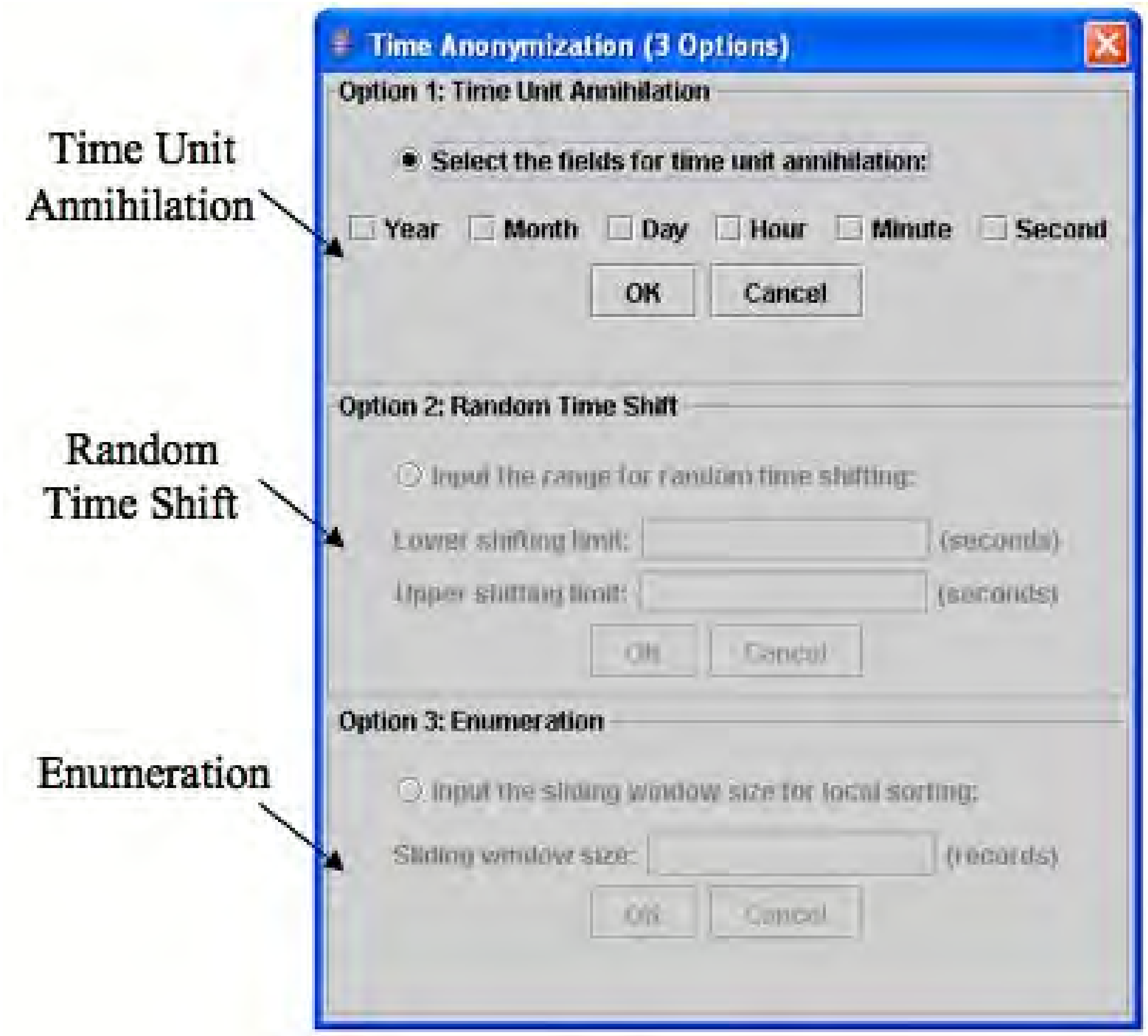
Option 3: Prefix-preserving pseudonymization

Input a passphrase (Note: your passPhrase will not be saved in CANINE)

(Recommend: use seed larger than 6 characters)

Ok Cancel

Timestamp Anonymization



Time Anonymization (3 Options)

Option 1: Time Unit Annihilation

● Select the fields for time unit annihilation:

☐ Year ☐ Month ☐ Day ☐ Hour ☐ Minute ☐ Second

OK Cancel

Option 2: Random Time Shift

○ Input the range for random time shifting:

Lower shifting limit: (seconds)

Upper shifting limit: (seconds)

OK Cancel

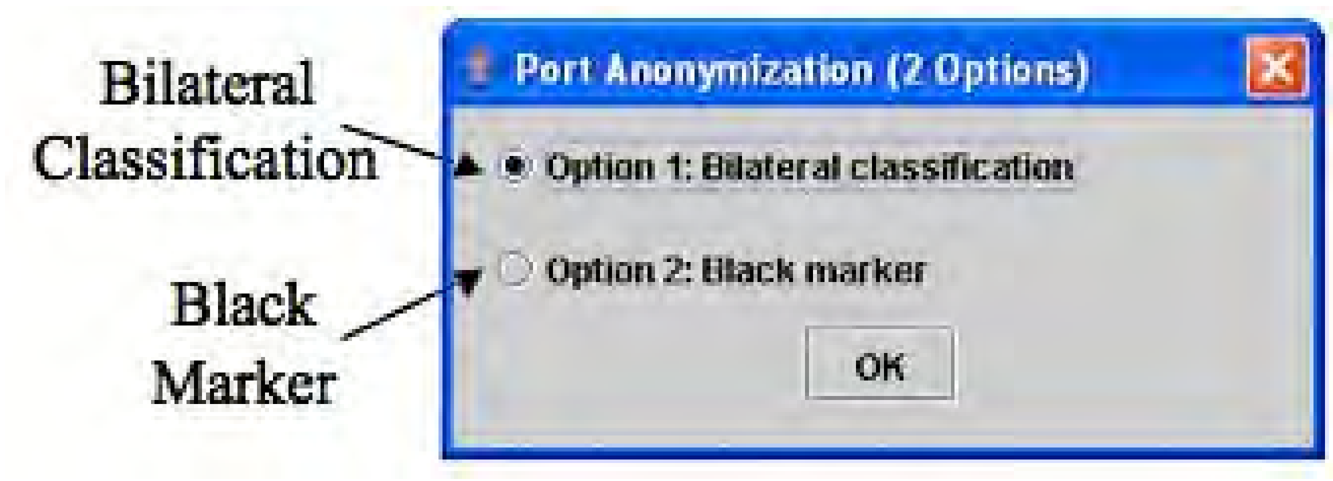
Option 3: Enumeration

○ Input the sliding window size for local sorting:

Sliding window size: (records)

OK Cancel

Port Number Anonymization


Bilateral Classification
Black Marker
Port Anonymization (2 Options)
Option 1: Bilateral classification
Option 2: Black marker
OK

•Bilateral classification

–Decide the port is ephemeral or not

•Black marker

CANINE: A NetFlows Converter/Anonymizer Tool for Format Interoperability and Secure Sharing

Katherine Luo, Yifan Li, Adam Slagell, William Yurcik
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
605 E. Springfield Avenue
Champaign, IL 61820
{*xluo1, yifan, slagell, byurcik*}*@ncsa.uiuc.edu*

Abstract

We created a tool to address two problems with using NetFlows logs for security analysis: (1) NetFlows come in multiple, incompatible formats, and (2) the sensitivity of NetFlow logs can hinder the sharing of these logs. We call the NetFlow converter and anonymizer that we created to address these problems CANINE: Converter and ANonymizer for Investigating Netflow Events). This paper demonstrates the use of CANINE in detail.

1 Introduction

A *network flow* is defined as a sequence of IP packets that are transferred between two endpoints within a certain time interval, and the most commonly used NetFlows formats are Cisco [1] and Argus [3]. With the increased use of NetFlows for network security monitoring [5], more and more tools based on NetFlows are being built and deployed. However, the different NetFlow sources, as well as collectors deployed, generate different incompatible versions of NetFlows. The different NetFlow formats impede the progress of network security monitoring since most tools that are based on NetFlows support only one format, but organizations often have hardware generating multiple formats. We were motivated to develop the CANINE to augment our existing flow tools [6, 7] by enabling them use NetFlows from the multiple sources here at the NCSA.

In addition to issues with format conversion, people often have concerns about information disclosure when sharing NetFlow logs—a source of sensitive network information. Consequently, we integrated anonymization capabilities with the converter. CANINE supports the anonymization of 8 fields common to all NetFlow formats: source IP address, destination IP address, starting timestamp, ending timestamp, source port, destination port, protocol and cumulative byte count. This combined converter and anonymizer has been vital to the development of visualization tools at the NCSA[6, 7] as it allows students to work with sensitive log data. We expect this work to likewise promote better insight into the use of NetFlows for security and network performance monitoring at other institutions.

The rest of the paper is organized as follows. Section 2 illustrates the system architecture of the CANINE. In section 3, we describe the the supported NetFlow conversion and anonymization methods. We conclude in Section 4.

2 System Architecture

The system architecture of CANINE is shown in Figure 1.

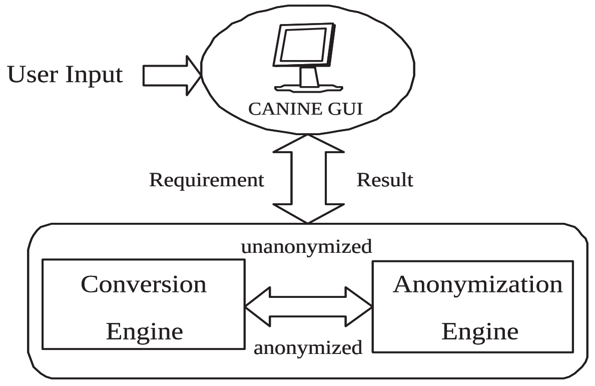
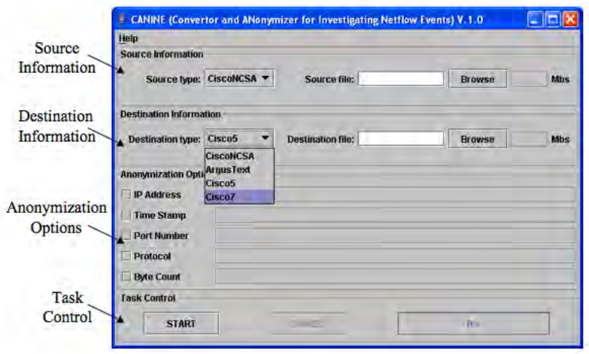


Figure 1: System Architecture of CANINE

CANINE consists of the two main modules: (1) the CANINE GUI and (2) the conversion/anonymization engines. The CANINE GUI accepts user input for NetFlow conversion and anonymization options, sends the request to the processing engine and summarizes the results of the performed actions in a pop-up window. First, the conversion engine reads the NetFlow data record from the source file and parses it into its component fields. Next it sends the unanonymized data to the anonymization engine. The anonymization engine houses a collection of anonymization algorithms, and it anonymizes the data according to the user's chosen options before it sends the data back to the conversion engine. The conversion engine reassembles the anonymized data according to the conversion options and writes the records to the destination file. Statistics are collected and sent back to the GUI which displays them in a new window.

3 Demonstration of CANINE

The root window of CANINE is shown in Figure 2. In the source [destination] information fields, the user can designate the source [destination] NetFlow format and file. Below these fields, the user can choose the fields to anonymize and the specific anonymization algorithms to use—many fields have multiple anonymization options. Below that, the *task control* area is used to start [stop] anonymization and display the current progress. CANINE can be freely downloaded at: **http://security.ncsa.uiuc.edu/distribution/CANINEDownLoad.html**


CANINE (Convertor and ANonymizer for Investigating Netflow Events) V.1.0

Source type: CiscoNCSA

Destination type: Cisco5

Figure 2: Main GUI of CANINE

3.1 NetFlow Conversion

CANINE's conversion engine currently supports conversion between Cisco V5, Cisco V7, Argus and NCSA unified formats. We briefly describe those formats below.

A. Cisco NetFlows

A *Cisco NetFlow* [2] record is a *unidirectional* flow identified by the following unique keys: source IP address, destination IP address, source port, destination port and protocol type. There are multiple versions of Cisco NetFlows (e.g., V1, V5, V7, V8 and V9). In all versions, the datagram consists of a header and one or more flow records. Most importantly, the header contains the version number and the number of records that follow in the datagram. For more details about the formats of each version, readers are referred to [1]. Currently, CANINE supports the most commonly used Cisco versions: V5 and V7. It will also support a mixture of V5 and V7 datagrams from an input file, though the output will all be in one format.

B. Argus NetFlows

Argus [3] views each network flow as a *bidirectional* sequence of packets that typically contains two sub-flows, one for each direction. Each flow record contains the attributes of source IP, source port, destination IP, destination port, protocol type, etc. There are two types of Argus records: the *Management Audit Record* and the *Flow Activity Record*, where the former provides information about Argus itself, and the latter provides information about specific network flows that Argus has tracked. For more details about the format, readers are referred to [4]. Note that unlike Cisco formats, Argus flows are ASCII text, rather than binary.

C. NCSA Unified Format

Since different versions of Cisco NetFlow Export datagrams are generated by the diverse routing equipment at the NCSA and because Cisco datagrams are of variable length, we have created the fixed length *NCSA Unified format* for use by our visualization tools ([6, 7]). This is important for efficiently supporting random access to records. The NCSA unified format contains the principle information about a network flow, as illustrated in Table 1 and serves as an internal format into which multiple versions of NetFlows can be transformed.

Table 1: NCSA unified record format

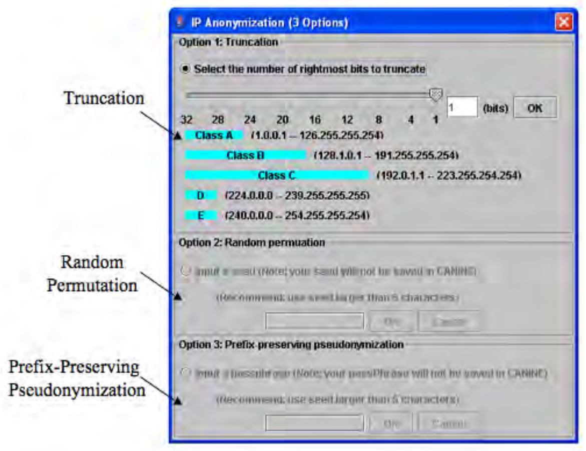
Data Field	**Length**(B)
version of Cisco NetFlow	1
padding (set to 0)	1
router's IP address	4
source IP address	4
destination IP address	4
source port number	2
destination port number	2
number of bytes	4
number of packets	4
protocol	1
TCP flags	1
start time (seconds since epoch)	4
milliseconds offset of start time	2
end time (seconds since epoch)	4
milliseconds offset of end time	2
padding (set to 0)	4

3.2 NetFlow Anonymization

CANINE's anonymization engine supports the anonymization of 8 data fields—only 5 unique types. Below we describe the anonymization options and their use within the latest version of CANINE.

3.2.1 IP Anonymization

We support three options to anonymize IP addresses. Note that either both the source and destination IP addresses are anonymized or both are unanonymized. You cannot anonymize one without the other. The IP anonymization options GUI is shown in Figure 3.



IP Anonymization (3 Options)

Option 1: Truncation

● Select the number of rightmost bits to truncate

32 28 24 20 16 12 8 4 1 — 1 (bits) OK

Class A (1.0.0.1 – 126.255.255.254)
Class B (128.1.0.1 – 191.255.255.254)
Class C (192.0.1.1 – 223.255.254.254)
D (224.0.0.0 – 239.255.255.255)
E (240.0.0.0 – 254.255.255.254)

Option 2: Random permuation

○ Input a seed (Note: your seed will not be saved in CANINE)
(Recommend: use seed larger than 5 characters)
OK Cancel

Option 3: Prefix preserving pseudonymization

○ Input a passphrase (Note: your passPhrase will not be saved in CANINE)
(Recommend: use seed larger than 5 characters)
OK Cancel

Figure 3: IP Anonymization Options

A. Truncation

For IP address truncation, the user chooses the number of least significant bits to truncate. For example, truncating 8 bits would simply replace an IP address with the corresponding class C network address. Truncating all 32 bits would replace every IP with the constant *0.0.0.0*.

B. Random Permutation

We also support anonymization by creating a random permutation seeded by user input. We implement this algorithm through use of two hash tables for efficient lookup. Note that the use of tables means that the permutation will be different for two different logs anonymized at different times.

C. Prefix-preserving Pseudonymization

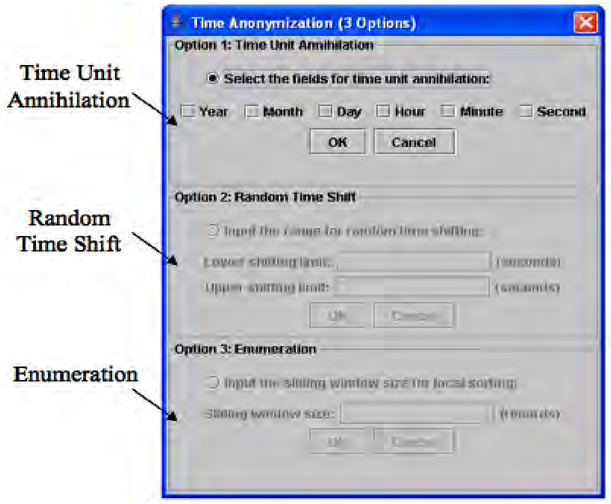
Prefix-preserving pseudonymization is a special class of permutations that have a unique structure preserving property. The property is that two anonymized IP addresses match on a prefix of n bits if and only if the unanonymized addresses match on n bits. We implemented the Crypto-PAn algorithm [8] for this type of anonymization, adding a key generator that takes a passphrase as input.

3.2.2 Timestamp Anonymization

Timestamps can be broken down into the units of *Year, Month, Day, Hour, Minute and Second*. We currently support three options to anonymize timestamps. The timestamp anonymization GUI is shown in Figure 4.

A. Time Unit Annihilation

We support the annihilation of any subset of the previously mentioned units. The user selects the time units to zero-out. For example, if someone wishes to obfuscate the date, they can remove the year, month and day information from the ending times. If they want to completely eliminate time



Time Anonymization (3 Options)

Option 1: Time Unit Annihilation

● Select the fields for time unit annihilation:

☐ Year ☐ Month ☐ Day ☐ Hour ☐ Minute ☐ Second

OK Cancel

Option 2: Random Time Shift

○ Input the range for random time shifting:
Lower shifting limit: (seconds)
Upper shifting limit: (seconds)
OK Cancel

Option 3: Enumeration

○ Input the sliding window size for local sorting:
Sliding window size: (records)
OK Cancel

Figure 4: Timestamp Anonymization Options

information, they can select all of the time units for annihilation. Start times are adjusted so that the duration of the flow is kept the same.

B. Random Time Shift

In some cases it may be important to know how far apart two events are without knowing exactly when they occurred. For this reason, a log or set of logs can be anonymized at once such that all timestamps are shifted by the same random number. The user needs to designate the lower and upper shift limit, from which the random number of seconds is generated. If one uses this type of anonymization on two different log files at different times, then this random number will be different between the data sets. We warn users to be aware of the troubles with data mining (by indexing the timestamp) between sets anonymized at different times in this manner.

C. Enumeration

With this method, all time information is essentially removed except the order in which the events occurred. A random end time for first record is chosen, and all other records are equidistantly spaced from each other—temporally that is—while retaining the original order with respect to ending times. Start times are adjusted so that the duration of the flow is kept the same. Sorting cannot work perfectly on streamed data, and it would be extremely slow on large log files. So we make use of the fact that records come roughly sorted by ending times and sort locally. This has worked with great accuracy and efficiency.

3.2.3 Port Anonymization

We support two anonymization options for port numbers. The port number anonymization GUI is shown in Figure 5.

A. Bilateral Classification

Usually, the port number is useless unless one knows the exact value to correlate with a service. However, there is

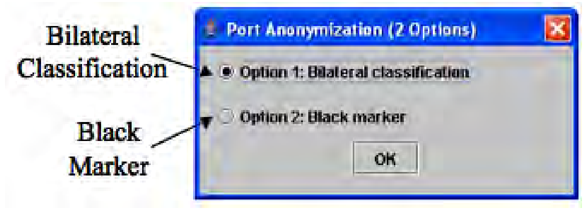

Bilateral Classification
Black Marker
Port Anonymization (2 Options)
Option 1: Bilateral classification
Option 2: Black marker
OK

Figure 5: Port Anonymization Options

one important piece of information that does not require one to know the actual port number: whether or not the port is ephemeral. In this way, we can classify ports as being below 1024 or greater than 1023. To keep the format the same for log analysis tools, port 0 replaces all ports less than 1024, while port 65535 replaces the rest of the port values.

B. Black Marker Anonymization
From an information theoretic point of view, this method is no different than printing out a log and blacking-out every port number. In a digital form, we just replace all ports with a 16 bit representation for 0.

3.2.4 Protocol Anonymization

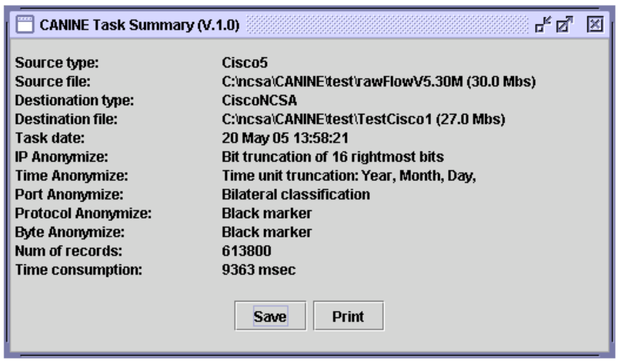
Protocol information can be eliminated with CANINE. We do this by replacing the protocol number with the unused, but IANA reserved, number of 255. This is the maximal number for that 8 bit field.

3.2.5 Byte Count Anonymization

For user privacy, one may desire to eliminate byte counts. Thus we support black marker anonymization of this field. All byte counts are replaced with the constant of 0, an impossible byte count in reality because headers do account for some of those bytes.

3.3 Task Result Dialog

After the CANINE task finishes, a brief task summary will be shown to the user in a pop-up window (Figure 6).



Source type:	Cisco5
Source file:	C:\ncsa\CANINE\test\rawFlowV5.30M (30.0 Mbs)
Destionation type:	CiscoNCSA
Destination file:	C:\ncsa\CANINE\test\TestCisco1 (27.0 Mbs)
Task date:	20 May 05 13:58:21
IP Anonymize:	Bit truncation of 16 rightmost bits
Time Anonymize:	Time unit truncation: Year, Month, Day,
Port Anonymize:	Bilateral classification
Protocol Anonymize:	Black marker
Byte Anonymize:	Black marker
Num of records:	613800
Time consumption:	9363 msec

Figure 6: Task Summary Dialog of CANINE

The task summary includes the following information: source and destination formats/filenames, date of processing, anonymization methods used, number of records processed and the total processing time. The user can save and print the task summary for future reference.

4 Summary

In this paper, we put forth two important problems facing the developers of NetFlow based tools: (1) NetFlows come in different and incompatible formats, and (2) the sensitive nature of NetFlow logs make it difficult for developers to find good data sources. Our tool, CANINE, addresses both of these issues by giving users the ability to both convert and anonymize NetFlow logs.

While users have many options to anonymize NetFlows with CANINE, it can still be difficult to choose the correct options for a particular organization's needs. Thus, future work should focus on creating multiple, useful levels of anonymization that trade-off between the utility of the anonymized log and the security of the anonymization scheme. This work should also strive to help organizations map levels of trust shared with would-be receivers to these different levels of anonymization.

References

[1] "Cisco NetFlow Services and Applications White Paper", Jun 2005; <http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm>.

[2] K. Claffy, G. C. Polyzos and H. W. Braun. "Internet traffic flow profiling", UCSD TR-CS93-328, SDSC GA-A21526, 1993.

[3] C. Bullard, "Argus, the network Audit Record Generation and Utilization System", June 2005; <http://www.qosient.com/argus/>.

[4] C. Bullard, "Argus record format", June 2005; <http://www.qosient.com/argus/argus.5.htm/>.

[5] Yiming Gong, "Detecting Worms and Abnormal Activities with NetFlows", August 2004; <http://www.securityfocus.com/infocus/1796>

[6] K. Lakkaraju, W. Yurcik, A. Lee, R. Bearavolu, Y. Li and X. Yin, "NVisionIP: NetFlow Visualizations of System State for Security Situational Awareness. VizSEC/DMSEC, held in conjunction with 11th ACM Conference on Computer and Communications Security, Fairfax, VA, October 2004.

[7] X. Yin, W. Yurcik, M. Treaster, Y. Li and K. Lakkaraju, "VisFlowConnect: NetFlow Visualizations of Link Relationships for Security Situational Awareness", VizSEC/DMSEC, held in conjunction with ACM Conference on Computer and Communications Security, Fairfax, VA, October 2004.

[8] A. Slagell, J. Wang and W. Yurcik, "Network Log Anonymization: Application of Crypto-PAn to Cisco NetFlows, Secure Knowledge Management Workshop, Buffalo, NY, 2004.

Correlations between quiescent ports in network flows

Joshua McNutt
Markus De Shon
CERT Network Situational Awareness Group,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
{jmcnutt,mdeshon}@cert.org

Abstract

TCP/IP ports which are not in regular use (quiescent ports) can show surges in activity for several reasons. Two examples include the discovery of a vulnerability in an unused (but still present) network service or a new backdoor which runs on an unassigned or obsolete port. Identifying this anomalous activity can be a challenge, however, due to the ever-present background of vertical scanning, which can show substantial peak activity. It is, however, possible to separate port-specific activity from this background by recognizing that the activity due to vertical scanning results in strong correlations between port-specific flow counts. We introduce a method for detecting onset of anomalous port-specific activity by recognizing deviation from correlated activity.

1 Introduction

The CERT Network Situational Awareness Group is using SiLKtools [1] to analyze Cisco NetFlow data collected for a very large network. Details on the functionality of SiLKtools can be found in other publications from our center. [1]

The analysis techniques in this paper can be used on any flow-based data source. In our case, the analysis is performed on hourly summaries on the inbound number of flows, packets and bytes on each port, where "inbound" simply refers to traffic coming into the monitored network from the rest of the Internet.

Analysis of network flows for anomalous traffic can be challenging due to fluctuations in traffic that are resistant to variance-based statistical analysis. [2] We have discovered that, for a restricted set of network phenomena, correlations between flow counts on different ports can be a useful way of filtering out "background" activity.

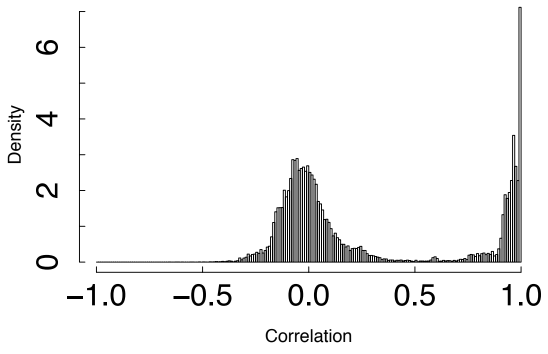


Figure 1: Histogram of robust pairwise correlation values (not including self-correlations or duplicates, since the correlation matrix is symmetric) for hourly flow counts on server ports 0-1024. Note that a significant proportion of the ports are well-correlated.

2 Correlations

Our data shows extremely high correlation (frequently > 0.99) between flow counts on many ports which do not have active services running on them (see Fig. 1). Because of the lack of "normal" traffic on these ports, any activity which is present is very likely to be due to vertical port scanning. As long as the port scanning proceeds quickly enough, then a vertical scan deposits the same number of flow records for each port within the time period over which flow counts are summed. Thus, the number of flows to each port will be highly correlated with the number of flows to other quiescent ports. This characteristic is indeed observed on the very large network that we are monitoring.

Given that a set of ports are normally correlated, then by calculating the median of the number of flows on each port in an hour, and then subtracting that median value from the number of flows observed on each port in that hour, we can remove the correlated background activity from analysis. This background-subtracted time series can then be analyzed for port-specific behavior through normal peak-finding algorithms.

A useful (though untested) method for detecting the remaining peaks might include using a "trimmed mean" (mean calculated from the data points remaining after removing outlier data points) and "trimmed standard deviation." The "trimmed" mean and standard deviation would be used similarly to the ordinary mean and standard deviation to identify outliers (flow counts which lie above the mean by some multiple of the standard deviation).

The use of the median instead of the mean, and the use of "trimmed" means and standard deviations in our method is for the same reason we used a "robust" correlation method as described below–to prevent outliers (which would be the things we are trying to detect) from attenuating the sensitivity of the detection algorithm inappropriately.

2.1 Correlation clusters

If traffic on port A is correlated with port B, and port B is correlated with port C, then port A is also correlated with port C. Thus, ports A, B and C will form a cluster of correlated ports. We processed our correlation matrices to discover such clusters of ports which are all mutually correlated. These clusters are surprisingly large, and could be even larger in situations (unlike our own) where traffic is not filtered (a darknet, for example).

To prevent isolated anomalies during the learning period from interfering with identifying the true clusters, we used a "robust correlation." The robust correlation measure is calculated using the minimum volume ellipse approach. This method was discussed in [3] in the context of calculating robust statistical distances. Since correlation is a measure which is highly sensitive to even one or two outliers, we wish to exclude extreme observations. Therefore, the data used for the correlation calculation consist of all points enclosed by the 95 percent minimum volume ellipse. This is the smallest possible ellipse which covers 95 percent of our data.

For incoming traffic on TCP ports 0-1023, using the "robust" correlation measure, and requiring a correlation ≥ 0.96 for one port to be considered correlated with another port, we found a single cluster of 133 ports, and a second cluster of 3 ports. More careful analysis may reveal the clusters of mutually correlated ports to be larger, if some ports had sustained anomalous activity, but are otherwise well-correlated.

3 Server ports

The ports numbered 0-1023 are by convention reserved for use by server programs owned by the "superuser" or system user. For this reason, the traffic patterns on these ports are quite different than for the higher-numbered "ephemeral" ports. The traffic on our network is consistent with this generalization.

Since a number of ports in the server port range are in active use by common services (most notably "web" traffic on ports 80/tcp and 443/tcp, and "email" on port 25/tcp), and others are usually filtered on real-world networks, not all ports would be expected to correlate well. However, unused ports, whether unassigned or obsolete, would be expected to have very little active traffic; we call such ports "quiescent," i.e. mostly quiet.

Correlations on quiescent server ports arise from the presence of vertical scanning activity (where a source host is scanning through all port numbers, or at least the server port numbers, on a target host). Deviations from correlated activity would be expected in the case of horizontal scanning (scanning for a particular port across hosts). An onset of horizontal scanning on a particular port might be expected if a new vulnerability is announced in an obsolete, but still present, service.

Onset of sustained activity which deviates from prior correlation could indicate the presence of a worm (self-propagating exploit program) on that port.

4 Ephemeral ports

The ports numbered 1024 and above are used by user space programs, primarily as temporary ports for outgoing connections by client programs such as web browsers. While this convention has been blurred somewhat by peer-to-peer programs and other unprivileged servers, the model still holds for most ports.

In our data set, nearly all the ephemeral ports show strong correlations, with daily and weekly seasonal patterns (see Figs. 2 and 3). This is consistent with the rhythms of user-driven traffic, meaning that the data comes from user space client programs connecting out to servers. Because such a connection creates two flow records (one for the outbound connection, but another for the return traffic within the same TCP session), we see return traffic flows in our "incoming" data set. An analysis of ephemeral port traffic verifies this hypothesis, with most of the data (where the definition of "most" depends on the day and time of day) consists of traffic from source ports 80/tcp, 443/tcp and 25/tcp, in that order.

Future improvements to our flow record collection software will allow easy differentiation of true incoming flows vs. return traffic from outbound connections. For

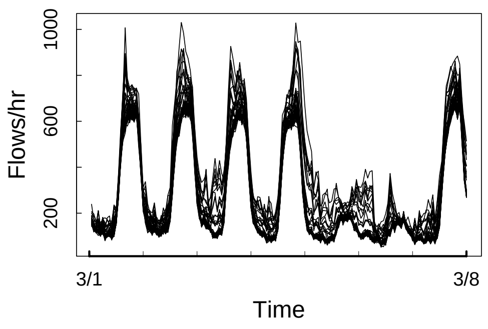


Figure 2: Example of incoming flow counts for 50 ephemeral ports for a one week time period in 2005. Note the daily and weekly seasonality consistent with user-generated activity.

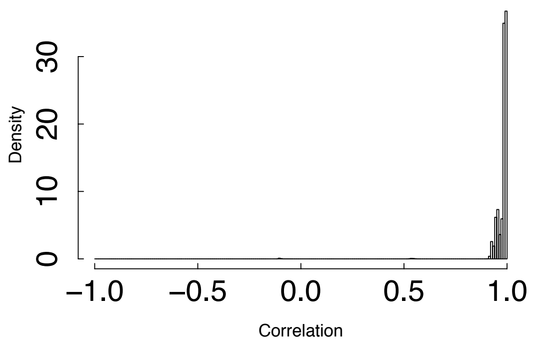


Figure 3: Histogram of correlation values for pairwise correlations between 1024 ephemeral ports (specifically, 50000-51024), excluding self-correlations and duplicates (since the correlation matrix is symmetric). Note the high concentration of highly correlated pairs.

the purposes of the analysis method described in this paper, however, the distinction is unimportant, as the return traffic patterns are highly correlated, as are any vertical scans taking place (though analysis has revealed that vertical scans of ephemeral ports are rare in our data). Deviation from correlated activity will have already removed the background of return traffic flows and vertical scanning, at least approximately. Any remaining significant peak activity will be due to special attention to a particular port or set of ports, just as with server ports.

Possible explanations for the onset of persistent deviant activity on an ephemeral port include: widespread scanning for a particular backdoor, port activity due to a new peer-to-peer protocol, the onset of activity for a worm that uses a particular ephemeral port to spread or perform other tasks, or scanning or exploit of a vulnerability in a mostly quiet server running on a high-numbered port.

5 Port 42/TCP, a case study

Port 42/TCP hosts the Microsoft Windows Internet Naming Service (WINS) service on Microsoft Windows hosts, an obsolete directory service which nevertheless was present in some versions of Windows as recent as Windows Server 2003, for backwards compatibility. On November 25, 2004, a remote exploit vulnerability in the WINS service was first announced by CORE Security Technologies (www.coresecurity.com) to their CORE Impact customers, in their exploits update for that day. The next morning, a more public announcement was made by Dave Aitel of Immunity, Inc. (www.immunitysec.com) on his “Daily Dave” email list. This vulnerability was later assigned CVE number CAN-2004-1080, and is discussed in CERT Vulnerability Note VU#145134.

In Fig. 4 we compare the data for incoming flow counts, destination port 42/TCP, to the median of incoming flow counts to several other ports. Fig. 5 shows the difference between the 42/TCP data and the median data. These two plots cover approximately a two month time period in 2004 preceding the announcement of the WINS vulnerability. The median value from a correlation cluster is used (rather than the mean) because a large deviation in one of the time series could significantly affect the mean, but not the median. The difference between a flow count and the median flow count for the cluster, therefore, would be a better indicator of the deviation from the expected value.

There are two significant periods of deviation in 42/TCP in the two month period before the announcement of the WINS vulnerability, which are explained below. The important thing to note is that the deviant peaks in the two week time period around October 15th, and the peak at October 28th, are well within the normal variability of the data. The correlation technique separates the background (due to vertical scanning) from the signal we are looking for (due to special attention to port 42, or port 42 and some list of other ports).

In early October, two IP’s scanned a set of 18 mostly non-contiguous ports (e.g. 22, 25, 53, 1080) on the monitored network, including port 42. Because of the larger number of ports targeted, these scans probably do not indicate foreknowledge of the WINS vulnerability to be announced the next month. Instead, the set of ports could have been used to determine active hosts, and a simple OS identification (port 42 indicating Microsoft Win-

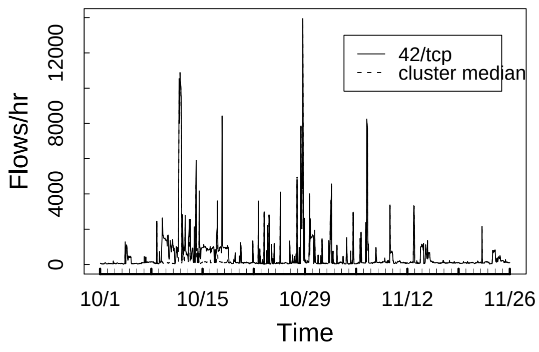


Figure 4: Incoming flow counts to destination port 42/TCP from our data and median incoming flow counts to several other destination ports. The dashed line showing the median flow counts is mostly obscured by the solid line because of the high correlation. An exception is the uncorrelated 42/TCP peaks in the two week period centered on October 15th (which are pictured more clearly in Fig. 5). Note that the uncorrelated peaks are within the normal variation of the activity.

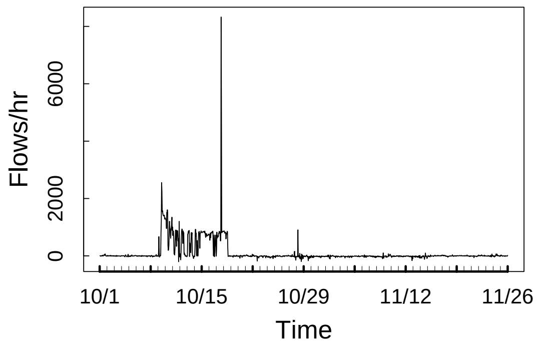


Figure 5: The difference between the two time series in Fig. 4. The two-week period of deviation, and the smaller isolated peak, are explained in the text.

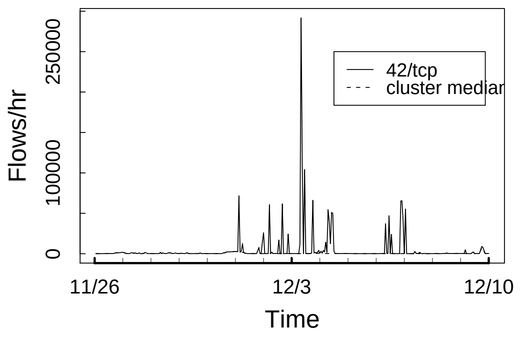


Figure 6: Incoming flow counts to port 42/TCP after the announcement of the WINS vulnerability, compared to median incoming flow counts to several other ports.

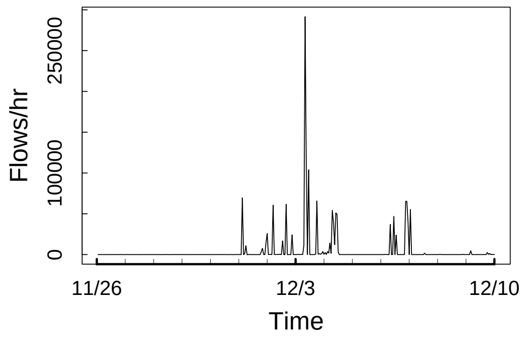


Figure 7: Difference between flow counts to port 42/TCP and the median flow count to other correlated ports, after the WINS vulnerability announcement.

dows, for example).

The smaller peak in late October appears on closer analysis to be benign activity, possibly due to some legacy systems attempting to use the WINS service.

While these these port 42-specific activities do not represent important security events, the fact that they were found easily using this method indicates that port-specific activity of a more malicious nature, which would otherwise be obscured by the background noise of vertical scanning in the server port range, could be discovered easily using the methods described in this paper.

The data after the WINS vulnerability announcement shows a significant peak in the number of incoming flows starting on December 1st at 2:00am GMT, but the number of hosts involved was still small. By midnight GMT of that same day, however, the number of hosts had surged considerably, and it would have been clear that there was new, widespread interest in port 42/TCP.

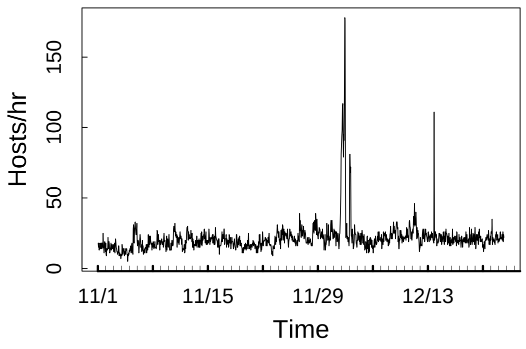


Figure 8: Number of unique hosts per hour attempting connections to 42/TCP from the Internet. By 12:00am GMT December 2nd the number of hosts was clearly much higher than had previously been seen.

The first public announcement we were able to find of widespread scanning on port 42/TCP was on December 13, in an email message by James Lay to the Full Disclosure email list—11+ days after significant scanning was clearly visible in our data using our correlation technique. If we had been using our correlation technique operationally at that time, an earlier announcement of widespread scanning would have been possible.

6 Port 2100/TCP

Port 2100/TCP lies in the "ephemeral" port range, but is actually also used for the Oracle FTP service. An exploit was released on March 18, 2005 for a vulnerability announced in August of 2003.[2] Fig. 9 clearly shows that scanning of port 2100/TCP commenced at that time.

7 Conclusions

Our analysis of port 42/TCP traffic shows a clear onset of scanning activity specific to port 42 after the announcement of the remote exploit vulnerability in the WINS service announced in late November of 2004. The scanning activity was clearly detectable well before any public announcement of such scanning.

The usefulness of subtracting the correlated background from per-port traffic summaries to detect port-specific behavior lies in the simplicity of the method, and in its ability to ignore vertical scanning as well as the background of web/email activity.

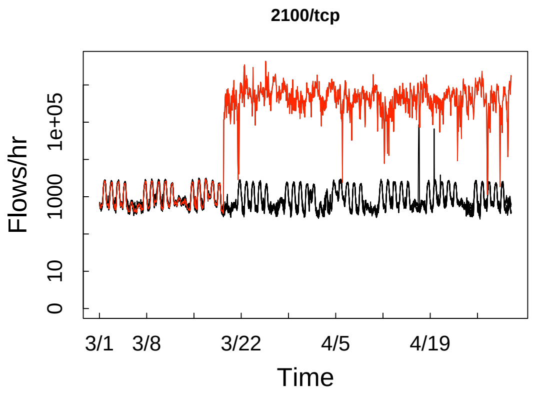


Figure 9: Flows per hour incoming to port 2100/TCP in March–April 2005 (red, or upper, line) as compared to nine ports which were correlated to 2100/TCP at the beginning of that time period.

References

[1] CARRIE GATES, MICHAEL COLLINS, E. A. More netflow tools: For performance and security. In *LISA XVIII* (2004), pp. 121–131.

[2] LELAND, W. E., TAQQ, M. S., WILLINGER, W., AND WILSON, D. V. On the self-similar nature of Ethernet traffic. In *ACM SIGCOMM* (San Francisco, California, 1993), D. P. Sidhu, Ed., pp. 183–193.

[3] ROUSSEEUW, P. J., AND VAN ZOMEREN, B. C. Unmasking multivariate outliers and leverage points. *Journal of the American Statistical Association 85* (1990), 633–639/648–651.

Notes

[1]http://silktools.sourceforge.net/
[2]http://www.oracle.com/technology/deploy/security/pdf/2003Alert58.pdf

R: A Proposed Analysis and Visualization Environment for Network Security Data

Joshua McNutt
CERT Network Situational Awareness Group,
Carnegie Mellon University, Pittsburgh, PA 15213, USA
jmcnutt@cert.org

Abstract

The R statistical language provides an analysis environment which is flexible, extensible and analytically powerful. This paper details its potential as an analysis and visualization interface to SiLK flow analysis tools as part of a network situational awareness capability.

1 Introduction

The efficacy of network security analysis is highly dependent upon the data interface and analysis environment made available to the analyst. The command line seldom offers adequate visual displays of data, while many GUI designs necessarily limit the query specificity afforded at the command line. This paper proposes the use of R, a statistical analysis and visualization environment, for interfacing with flow data. R is a complete programming language and, consequently, is highly extensible. Its built-in analysis and visualization capabilities provide the analyst with a powerful means for investigating and modeling network behavior.

2 R! What is it good for?

R is a language and environment for statistical computing and graphics used by statisticians worldwide. It is syntactically very similar to the S language which was developed at Bell Laboratories (now Lucent Technologies). Unlike S, R is available as free software under the terms of the Free Software Foundation's GNU General Public License in source code form. Additional details are provided by the R Project for Statistical Computing (http://www.r-project.org/). The website also provides links to documentation and program files for downloading. Supported platforms include Windows, Linux and MacOS X.

R is an object-based environment which can run interactively or in batch mode. It has the ability to generate publication-quality graphical displays on-screen or for hardcopy. Users can write scripts and functions which leverage the programming language's many features, including loops, conditionals, user-defined recursive functions and input/output facilities. For computationally-intensive tasks, C and Fortran code can be linked.

There are a handful of packages supplied with the R distribution covering virtually all standard statistical analyses. Many more packages are available through the Comprehensive R Archive Network (CRAN), a family of Internet sites covering a very wide range of modern statistical methods.

3 SiLK Tools

The suite of command line tools known as the System for Internet-Level Knowledge (SiLK) are used for the collection and examination of Cisco NetFlow version 5 data. The CERT Network Situational Awareness (NetSA) Team wrote SiLK [1] for the purpose of analyzing flow data collected on large volume networks. Flow data provides summaries of host communications providing a comprehensive view of network traffic.

The SiLK analysis tools provide Unix-like commands with functionality that includes selecting (a.k.a. filtering), displaying (ASCII output), sorting and summarizing packed binary flow data. Multiple commands can also be piped together for complex filtering. In this paper, we utilize the tools *rwfilter* (to select the data) and *rwcount* to generate binned time series of flow records, bytes and packets and feed the results into R for analysis. Further details on the functionality of SiLK can be found in [1].

4 Motivation: Command Line versus GUI

Many experienced users enjoy the query specificity afforded by the command line. But, in order to visualize

R Objects	
Object	**Description**
vector	ordered collection of numbers
scalar	single-element vector
array	multi-dimensional vector
matrix	two-dimensional array
factor	vector of categorical [2] data
data frame	matrix-like structures in which the columns can be of different types (e.g., numerical and categorical variables)
list	general form of vector in which the various elements need not be of the same type, and are often themselves vectors or lists. Lists provide a convenient way to return the results of a statistical computation.
function	an object in R which manipulates other objects

Table 1: Data object types in R

their data, they must make do with a third-party graphing program. They often do not favor a graphical user interface because their options for both queries and visualization tend to become more limited. What we hope to provide with the R interface is a preservation of command line control with the added features of integrated visualization and analysis. Essentially, we would describe it as an enhanced command line experience, but it also provides the analyst with all of the benefits of the R language's object-based workspace model.

5 R Data Manipulation

5.1 R Data Objects

Every entity in the R environment is an object. Numeric vectors, ordered collections of numbers, are the simplest and most common type of object, but there are many others. See Table 1 for a description of the object types.

In this paper, our example uses a data frame to store our data. The data frame object is a very flexible matrix-like entity which, unlike a matrix, allows the columns to be of different types.

5.2 SiLK Data Access

It should be noted that while we use R to interface with SiLK, virtually any command-line tool could be used with R. Also, R has multiple SQL database interface libraries. Many methods exist for interfacing with data stores. We detail below the R-SiLK interface being used at this time.

Within R, wrapper functions tied to specific tools in the SiLK suite read in the user-specified SiLK command line as a text-string parameter. The wrapper function makes a system call to the computer running the flow tools. Then, using a standard R data input function, the wrapper function reads in the ASCII output of the command line call. The results of the wrapper function call are assigned to a list object in R. Each element of that list represents a different analysis result, e.g. a matrix of the data, summary statistics, etc. Subsequent analysis and visualization operations can then be applied to that output object or any of its elements.

5.3 R Workspace

All objects are located in the user's workspace which can be saved at the conclusion of the R session and restored at the start of the next session. The command *history()* produces a list of all commands submitted to R by the user.

5.4 Analysis Capability

From simple summary statistics to advanced simulations, the R platform provides functions, extension packages (available through CRAN) and visualization capabilities appropriate to a wide range of flow analysis tasks. The object-based nature of the R environment makes it a useful platform for the network security analyst. Objects from different analyses can be preserved in the user's workspace for comparison purposes. Also, rapid prototyping of new analysis tools is possible due to the wealth of built-in capabilities and the ease with which new functions can be written.

The CERT/NetSA Team has used R for a variety of analysis tasks, from logistic regression to robust correlation analysis. We have used its SQL interface functionality to access hourly roll-ups of flow data summarized by port and protocol from a special database created specifically for port analysis. This has made it possible to study temporal correlations in port activity and identify ports which are exhibiting substantial volumetric changes.

5.5 Graphing Capability

One of the most important features of R is its ability to create publication quality graphical displays. R has a huge set of standard statistical graphs, stemplots, boxplots, scatterplots, etc. Extension packages are available for more advanced 3D plotting and highly-specialized display types. The advantage for the analyst running R in interactive mode is the ability to make slight changes

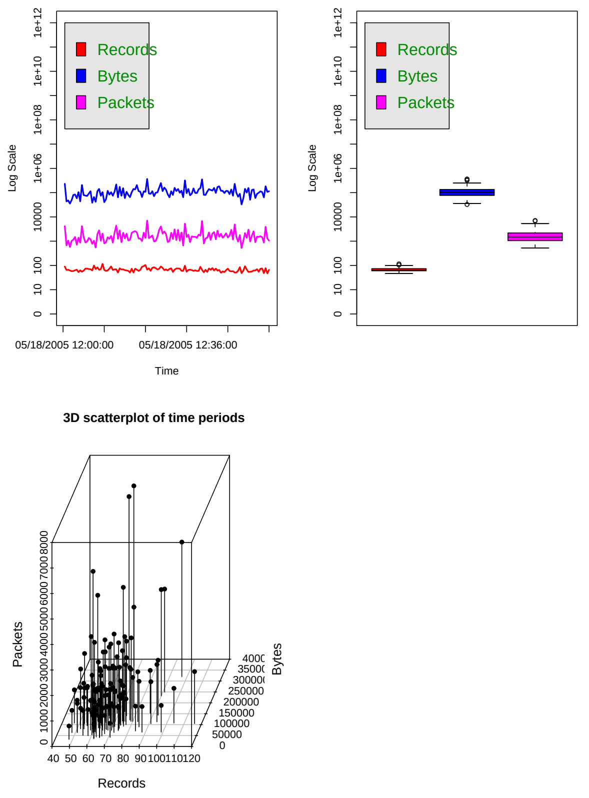


Figure 1: Graphical output of rwcount.analyze()

to the SiLK query and quickly visualize those changes in a newly drawn graph. Given the flexibility of its graphical facilities, R is also an ideal environment for advanced analysts to perform visualization prototyping.

6 R-SiLK wrapper function prototype: *rwcount.analyze()*

Our first proof-of-concept SiLK interface function is the wrapper *rwcount.analyze()* which calls the SiLK tool *rwcount*. Details of this wrapper function are provided in Table 2. The function has two input parameters, *command* and *plot*. The parameter command is a text string which is assigned a SiLK command line call to *rwcount*, which returns binned time series of records, bytes, and flows. The other input parameter, *plot*, determines whether a graphical display will be generated at runtime. The default is *plot*=TRUE. The visualization provided in our prototype includes three plots: a time series plot, side-by-side boxplots, and a 3D scatterplot of the data. Figure 1 provides an example of the graphical output generated by *rwcount.analyze()*.

When *rwcount.analyze()* is called, its output is assigned to a list object in R. The list it generates contains five elements: *data, command, stats, cor,* and *type*. These elements are defined in Table 2.

A sample R session using *rwcount.analyze()* to examine FTP traffic is provided below. The parameter command is assigned a SiLK command line. In our example, we specify TCP traffic (−−proto=6) directed at destination port 21 (−−dport=21) for the hour between noon and 1 p.m. on May 18, 2005. Those specifications are provided to *rwfilter* via switches, and the selected flows (in binary, packed format) are piped into *rwcount* where we have specified a bin size of thirty seconds (−−bin-size=30). The output of *rwcount* consists of the time series of bytes, records and packets which are read into a data frame object in R. This data frame is also an element in the output list object returned by *rwcount.analyze()*.

In this example, the output list returned by the function is assigned to *obj*. The list of object elements are printed with the function *names()* and correspond to the items in Table 2. As an example of automated analysis that can be returned in a results object, the correlation matrix of the series is found in *obj$cor*. This output shows that bytes, records and packets are highly correlated with each other ($\rho > .99$). Since *obj$data* is a data frame of the three time series, we can print the records field by typing *obj$data$Records*. This is one of the time series plotted in Figure 1.

```
> obj <- rwcount.analyze(command=
"rwrun rwfilter
--start-date=2005/05/18:12:00:00
--proto=6
--dport=21
--print-file
--pass=stdout |
rwcount
--bin-size=30",
plot=TRUE)

> names(obj)
[1] "data" "command" "stats" "cor"
[5] "type"

> obj$cor
             Records   Bytes    Packets
Records     1.0000   0.9944   0.9951
Bytes       0.9944   1.0000   0.9964
Packets     0.9951   0.9964   1.0000

> obj$data$Records
                          Records
05/18/2005 12:00:00    76218
05/18/2005 12:05:00    73374
```

rwcount.analyze() details	
Input Parameters	
Parameter	**Description**
command	SiLK command line text string
plot	Logic element determines whether R will perform runtime plotting
Output List Elements	
List Element	**Description**
data	Data frame containing rwcount time series for Bytes, Records and Packets
command	Same as input parameter description
stats	Summary statistics for Bytes, Records and Packets
cor	Correlation matrix for Bytes, Records and Packets
type	Text string to indicate which wrapper function generated this object

Table 2: rwcount.analyze() function description

```
05/18/2005 12:10:00   55743
...
```

7 Analyst Benefits

One of the advantages of R is its potential for rapid analysis prototyping. A user can very quickly write functions that generate a slew of experimental analysis results describing a host, a subnet, or traffic volumes. Each result can be included in the function's output list and evaluated. Analysis results which prove useful can be quickly integrated and become standard output elements.

In analytical work, the ability to label preliminary results objects provides the investigator with a facility for generating an audit trail. In R, this labeling is performed by the addition of object elements which describe the object to either the analyst or other functions which will operate on the object. By default, *rwcount.analyze()* returns the elements *type* and *command*. The element *type* can be used to describe the object to other functions. For example, a generic graphing function (perhaps called *rw.visualize()*) would read in an object and determine how it should be displayed based upon its *type*. The element *command* describes to the user how the object was created by storing the SiLK command. Additional elements can also be added to existing objects. For instance, a user may wish to attach a comment (e.g. "Surge in host count lasted for 6 hours") to an object by adding a text string element.

Since objects are preserved when the users save their workspace in R, comparison with objects from future analyses is very simple. Also, the user can graph objects from a previous analysis side-by-side with new results.

We believe the experienced analyst will leverage the enhanced command line experience, fast visualization and rapid analysis prototyping. For analyses requiring longer data pulls, R can also serve as an integrated scripting and analysis environment.

We envision a hierarchy of analysis functions. At the lowest level would be functions like *rwcount.analyze()* which use a SiLK command line call as a parameter. A function at the next level of the hierarchy would allow a user to specify criteria of interest via function parameters (e.g. dport=80, proto=6). This function would both generate the necessary SiLK command line and submit it to *rwcount.analyze()* for processing. Using these functions, novice analysts unacquainted with the SiLK command line would be able to perform real analysis tasks immediately. These functions could also be used for learning purposes since the SiLK command line needed for the query is provided in the output object.

8 Future Work

Our wrapper function *rwcount.analyze()* is merely a proof-of-concept prototype of an interface between R and SiLK. Next steps include the development of additional wrapper functions, making further improvements to *rwcount.analyze()*, and developing a generic visualization scheme that reads the *type* field in an output object to determine the appropriate display.

9 Conclusion

This paper has introduced the reader to R, demonstrating an overlap between its capabilities and the needs of network security analysts. R provides a truly integrated environment for data analysis and visualization. Further, the ability to interface with SiLK flow analysis tools and other data storage formats makes it an ideal environment for enhancing and extending a network situational awareness capability.

References

[1] Carrie Gates, Michael Collins, e. a. More netflow tools: For performance and security. In *LISA XVIII* (2004), pp. 121–131.

Notes

1 http://silktools.sourceforge.net/

2 We are using "categorical" here to describe string character data (e.g. "male" versus "female").

NVisionIP: An Animated State Analysis Tool for Visualizing NetFlows

Ratna Bearavolu, Kiran Lakkaraju, William Yurcik

National Center for Supercomputing Applications (NCSA)

University of Illinois at Urbana-Champaign

Outline

- Motivation
- Situational Awareness & Visualization
- Visualization Criteria
- NVisionIP – Demo
- Conclusion

Motivation

- Motivated by the concerns of Security Engineers at NCSA
- How do you provide situational awareness of the network – awareness of the state of the devices on the network
- Focus on situational awareness then intrusion detection
- Wanted a tool where the user can **see** the state information of the devices on the network

Situational Awareness Using Visualization

- Use visualization to show information about the network
- Visualization is used because it is:
  - Easy to detect patterns in the traffic
  - Conveys a large amount of information concisely
  - Can be quickly created by machines
- Use the security engineers background knowledge and analysis capabilities along with the capability of machines to quickly process and display data.

Key Features of Network Visualizations for Security

- **Interactivity:** User must be able to interact with the visualization
- **Drill-Down capability:** User must be able to gain more information if needed
- **Conciseness:** Must show the state of the entire network in a concise manner

Interactivity

- Allow security engineer to decide what to see
  - Data views (Cumulative, Animation (interval lapse) and Difference)
  - Features to view (traffic in/out, number of ports used, etc)
  - Filtering

Drill-down capability

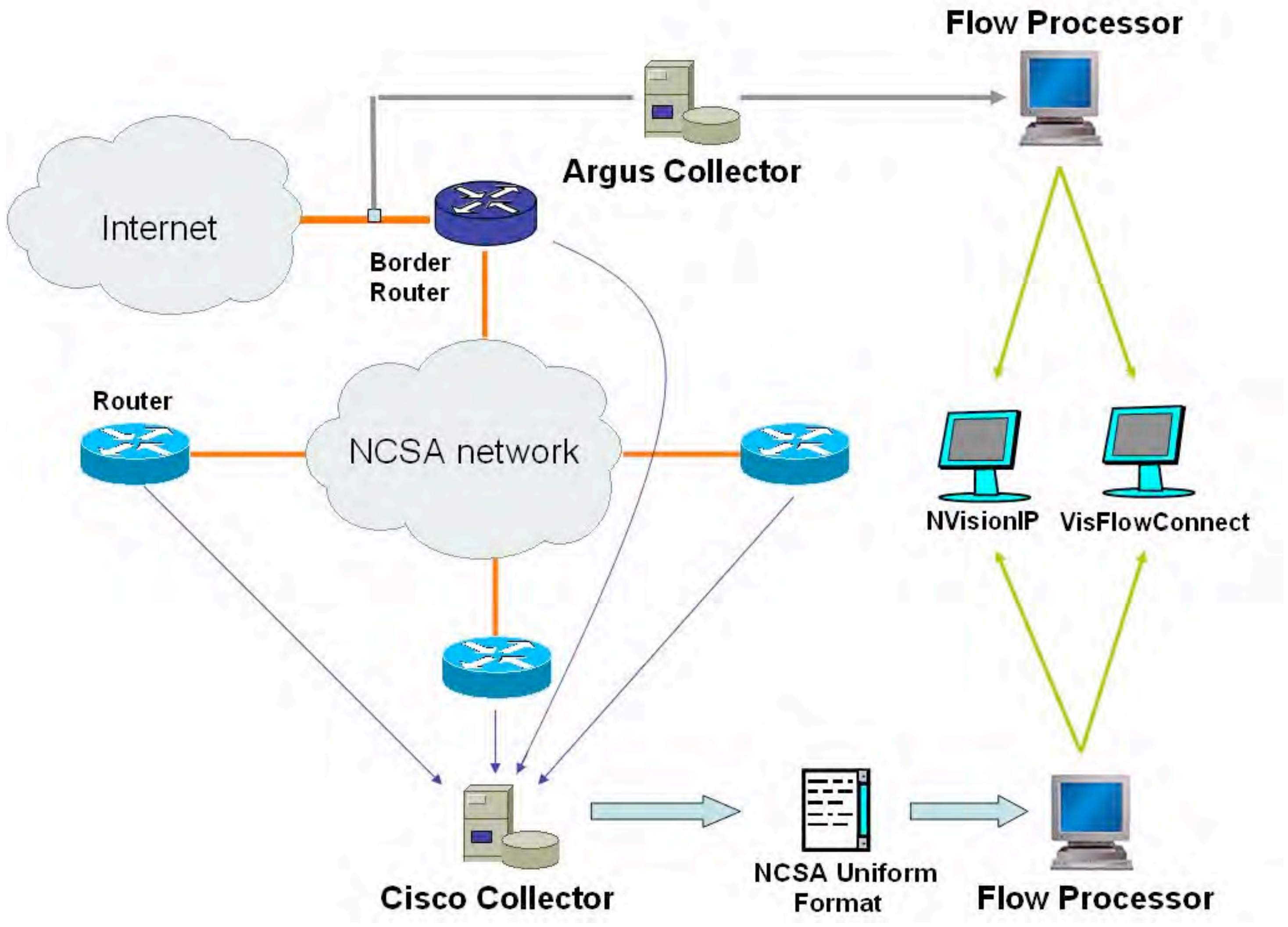
- Allow security engineer to see the network at different levels of resolutions
- Entire network – Galaxy View
- A subset of hosts – Small Multiple View
- A single machine (IP) – Machine View

Conciseness

- Allow a security engineer to view a large amount of information concisely
  - Show entire network with minimum of scrolling

…..thus allow security engineer to gain **situational awareness** of the network

Where is the data coming from at NCSA?



DEMO

For a single IP

- **FlowCount** - Number of times IP address was part of flow (Flow Count)
- **SrcFlowCount, DstFlowCount** – Number of time IP address was source and destination of a flow
- **PortCount** – Number of unique ports used
- **SrcPortCount, DstPortCount** – Number of unique ports used as source and destination ports
- **ProtocolCount** – Number of unique protocols used
- **ByteCount** – Number of bytes transferred.

Getting NVisionIP

- Distribution Website:

http://security.ncsa.uiuc.edu/distribution/NVisionIPDownLoad.html

- SIFT Group Website:

http://www.ncassr.org/projects/sift/

Conclusion

- Combine Security Engineers' skills with the visualization capabilities of machines.
- Visualizations with three key properties to provide Situational Awareness:
  - Interactivity
  - Drill-Down Capability
  - Conciseness

Questions

Flow Analysis and Interoperability: Data Models

Brian Trammell bht@cert.org
CERT Network Situational Awareness Group

The problem

Cooperative flow data analysis efforts are often hampered by incompatible native data formats among analysis tool suites.

Mandating a common format is impractical:

- Expensive to integrate into each suite.
- Least common denominator approach fails for suites which share uncommon information elements or data representations.

A solution

Translate flows and summaries at data sharing interface.

- Use native formats internally.
- “Single box” translation at the sharing interface avoids least common denominator issues.
- Modifying each flow at the sharing interface generally has to happen anyway, for sanitization and obfuscation purposes.

Flows as Events

“Event”: an assertion made by some event *source* that *something* happened at *some point in time*, possibly continuing for some *duration*.

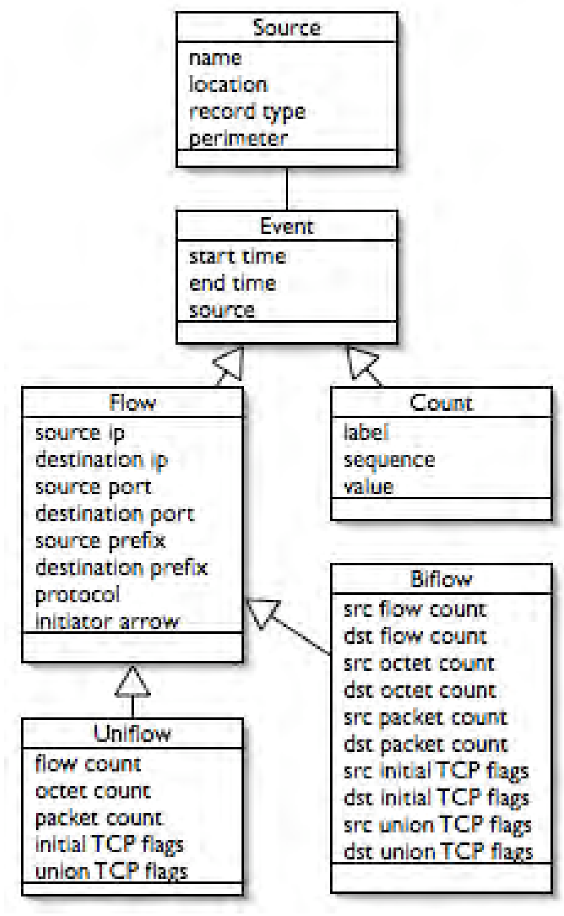
Event is “base class” from which all other classes of event data inherit.

Both raw flow records and many types of time-series analytical products can be represented as events.

Treating flows as events allows correlation with other (non-flow) data sources, as well.

- SIM/SEM
- NIDS/IPS

Event Data Model



Uniflows and Biflows

Raw Netflow data is unidirectional – one flow for each direction of a session (“uniflow”).

- necessary for asymmetric routing
- can be burdensome for analysis

Bidirectional flow data (“biflow”).

- sensing technologies which operate at L2 can generate biflows (e.g. Argus)
- matching uniflows into biflows possible, computationally expensive
- semantics of “source” and “destination” can become confusing

Associations

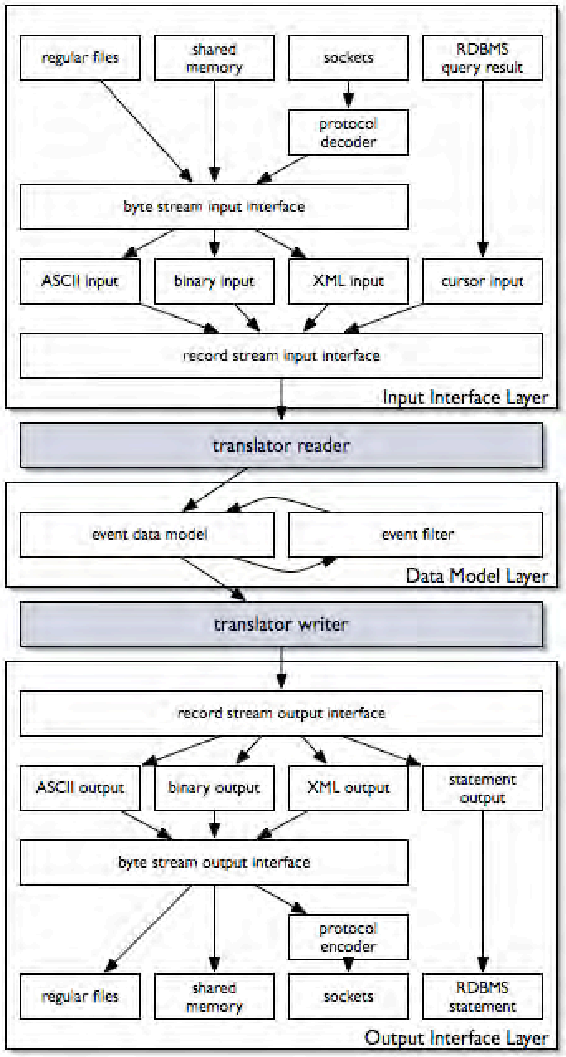
"Association": assertion made by some source that a set of "*key*" fields is known to *map* to a set of "*value*" fields, and that this mapping is known to be valid for a given *time range*.

Non-event data, useful for characterizing or aggregating events:

- Network
- Organization
- Country Code

Associations can be used for aggregation during translation, or may be translated themselves.

Proposed Translator Design



Incremental Development Plan

Current: NAF, libfixbuf

- modified event data model
- emphasis on accepting multiple raw flow file formats
- uses IPFIX as interchange format

Future: “Bender”

- full event data model
- full I/O abstraction layer
- “Single box” interchange

NAF

“NetSA Aggregated Flow”: reads from a variety of flow formats into a single biflow summary format based on IPFIX.

Allows aggregation of flows grouped by arbitrary fields in raw flow data.

Addresses issue of receiving raw flow data from multiple sources, but not of sharing summary data.

More compact than centralized storage and analysis of raw flows.

NAF native format can be manipulated by IPFIX-compliant implementations.

NAF Data Model

NAF uses a modified event data model.

- Event time replaced with aggregate time bin.

NAF aggregate flows can be represented by the full event data model.

- time bin ➔ start time
- bin length (+ time bin) ➔ end time

NAF Tools

nafalize

- aggregate raw flow data into NAF format.

nafscii

- print NAF formatted data as ASCII text.

nafilter

- select and/or sanitize NAF formatted data.
- not available in initial release.

Initial NAF tools public open source release in one-month timeframe.

IPFIX

IPFIX is an IETF protocol defining a template-driven, self-describing binary data format, and an extensible data model.

- Useful as a basis for defining new flow formats in an interoperable way.
- Information model can be extended to support other event types, flow summaries.
- Some gaps in built-in information model:
  - No support for bidirectional flows (biflows).
  - Single-record arrays are cumbersome (e.g. MPLS label stacks).

libfixbuf

IPFIX data format handling library.

- Handles templates, message and set headers.
- Transcodes data given two templates.
- Supports draft-trammell-ipfix-file-00 extensions for persistent storage of IPFIX formatted data.
- No protocol semantics, but could be used as basis for IPFIX exporting and collecting processes.

Used by NAF to implement its native format.

Available today:

- http://aircert.sourceforge.net/fixbuf

Questions?

A Proposed Translation Data Model for Flow Format Interoperability

Brian Trammell
CERT Network Situational Awareness Group
Carnegie Mellon University
Pittsburgh, PA 15213, USA
bht@cert.org

Abstract

A significant technical barrier to the growth of the security-oriented network flow data analysis community is the mutual unintelligibility of raw flow and intermediate analysis data used by the proliferation of flow data analysis tools. This paper presents a proposed solution to this problem, a common event data model and a translator built around it to adapt each tool's native format to this common model.

1 Introduction

While non-technical barriers do exist to collaborative network flow data analysis across administrative domains, and these barriers are in many cases formidable, organizations finding both the desire and the political will to share data in the pursuit of a greater awareness of activities on the Internet at large are soon presented with another problem; their tools will not talk to each other.

The Internet Engineering Task Force is presently addressing the raw flow data standardization problem with IPFIX (Internet Protocol Flow Information Export) [1], a flow data format and collection architecture based upon Cisco's NetFlow version 9. This standardization effort is a start. However, it it not a complete solution to the interoperability problem. IPFIX is a wire protocol designed to efficiently generate and move flow data from an observation point (such as a router) to a collector, and as such does not address issues of short- or long-term data storage, the expression of query results containing flows or flow data summaries, or the handling of ancillary data used in flow data analysis that is not necessarily flow-oriented itself.

In addition, mandating that each existing flow collection and analysis toolchain use IPFIX natively is not really a satisfactory answer. While these toolchains will have to support the import of IPFIX flow data as the IPFIX standard is deployed in new observation points, each of these formats has evolved for a reason. If this were not the case, little specialization would have occurred beyond raw Netflow itself.

How, then, to improve the technical state of interoperability and cooperation within the network flow data analysis community? This paper presents a proposed solution to this problem. Section 2 outlines the requirements of such a solution, sections 3 and 4 propose a data model meeting these requrements, and section 5 builds a candidate design for a translator around this model.

2 Requirements Analysis

We propose the application of an event translator to this interoperability problem. The translator's design must meet the following requirements in order to be useful:

- Universality: the translator must be able to handle any type of event, event summary, or associated data, and be able to translate between any two semantically compatible formats.
- Filterability: Since the interoperation point between organizations often requires data obfuscation or sanitization, the translator must support filtering during translation.
- Ease of Translation: the translator's design must seek to minimize the development time required to add a new known format to the translator.
- Performance: though it is too much to expect a workflow with a translation step to perform as well as one using a toolchain's native format, the translator must process data quickly enough to ensure that translation does not inordinately slow down the workflow.
- Portability: the translator must be able to run on multiple POSIX (or POSIX-like) environments to

reflect the variety of environments used for flow data processing.

3 Event Data Model

The key realization leading to this proposal is that at its core, all collections of security-relevant network data, whether from flow collectors, network intrusion detection and prevention systems, host monitoring and intrusion detection systems, security information managament products, etc., are made up of *events* . An event is simply an assertion that, according to some *event source* (e.g., a sensor, flow collector, etc.), something happened at some point in time, and possibly continued happening for some given duration. Therefore, every event record must have start and end timestamps and some identifier for the event source; these fields make up the *event core* data model.

Event records are made up of *key* and *value* fields. A key field is a field which defines some property of the event itself (e.g., packer header information), while a value field simply contains information about the event (e.g., counters). Each event is comprised of three or more key fields (including at least start, end, and source) mapping to zero or more value fields.

An event source is defined as a generator of a single *type* of event at a single network observation point, where type is defined by a list of key and value fields required of events of that type. This restriction is introduced to simplify processing of multiple types of data, e.g., flow events and NIDS alert events, collected at the same observation point; instead of associating a type with each event, the type is associated with the event's source.

3.1 Uniflows and Biflows

Raw flow data records contain an IP 5-tuple (source and destination IP address, source and destination port, and IP protocol), as well as packet and octet counters, and various routing information.

Flow data formats can be broadly split into two types: unidirectional (e.g., Cisco NetFlow V5), and bidirectional (e.g, Argus [1] and other pcap-based flow collection systems). We will call these *uniflows* and *biflows* for the sake of brevity. The requirement to handle both uniflows and biflows introduces some complexity into the semantics of source and destination.

For any uniflow, the meaning of source and destination are relatively straightforward; the source is the source IP address from the IP headers of the packets making up the flow, and the destination is the destination IP address from the headers. Biflows complicate the matter somewhat. The most straightforward way to assign source and destination for biflows is by the packet headers of the first packet observed, so ignoring instability at flow capture startup, the source address of a biflow corresponds to the connection initiator. However, we have seen at least one biflow format storing data differently than this; Q1Labs' QRadar [2] product presents flows by local and remote address, assuming some network perimeter under observation, and stores an additional "direction" arrow to note whether the connection is believed to have been initiated from inside or outside this perimeter. Note that this greatly confuses the definitions of "source" and "destination"; it may be better instead to refer two each address/port two-tuple as "endpoint A" and "endpoint B".

Free convertibility among these types requires that these semantics be stored for each event source, and that event sources storing biflows with an implicit perimiter must have a way of noting that perimeter.

Another difference between uniflows and biflows is that biflows have twice the counters than uniflows do; one counter of each type for packets initiated at each endpoint.

3.2 Summary Counters

It is also useful to exchange summary flow data. Retrospective traffic volume anomaly detection, for example, can be performed on flows aggregated by time bin and network. The core event data model natively supports time binning; an event with a start time at the beginning of the bin and an end time at the end of the bin represents a record for that bin.

For summarization purposes, the data model supports prefix lengths for source and destination address, as well as special "any" labels for other key fields.

TopN lists and other simple sequenced key/value counters bounded in time are another common type of summary data. The event data model supports these using multiple events, one for each place in the sequenced count list. Each of these events contains the key field and the count value, as well as a sequence number to provide order to the collection of events.

3.3 Event Classes

The event data model as described so far would seem to inherit from one another; that is, both flows and counts are types of events, and both uniflows and biflows are types of flows. Therefore, the data model can naturally be decomposed into classes in an object oriented design.

The classes comprising the event data model are illustrated in figure 1. The event data model is designed to be extensible, so new fields can be translated from one format to another, with the caveat that each class within the data model is immutable, and that extensions are

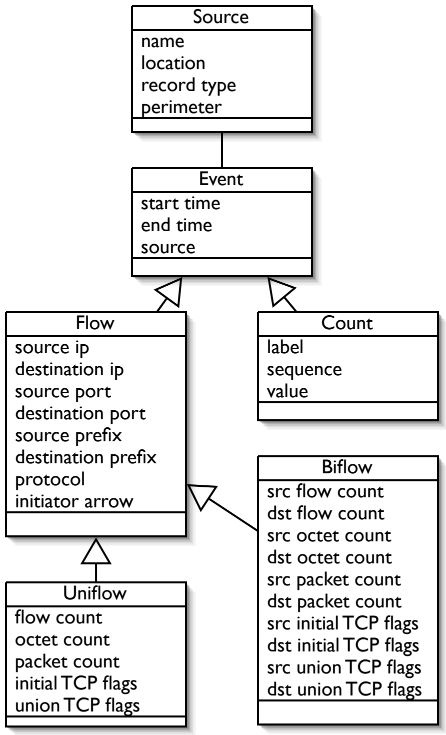


Figure 1: Event Model conceptual UML class diagram

achieved only by adding new classes which inherit from existing ones, not by adding fields to existing classes. This restriction is intended to reduce issues with data model object versioning at implementation time.

3.4 Other Event Types

Though the event core data model can handle any event placed in time and associated with an event source, this document is limited in scope to flow events and associated counters. Future revisions that deal in supporting correlations among event data types such as flow summaries and SIM events will extend the model accordingly.

3.5 Event Source Class

Event sources are handled by reference in the event data model. The one translation task presently supported by the event data model that requires information about the event source is translation between implicit-perimeter biflows and biflows without an implicit perimeter. The event source class therefore contains a perimeter field, which specifies the "local" network as a set of CIDR blocks or network address ranges.

4 Association Data Model

It is also useful to make assertions about the environment under monitoring that are not events, in order to further describe or aggregate events. An *association* is an assertion that, within a given time range, one set of key fields is known to map to another set of key and/or value fields. Associations have sources as do events; however, while association sources are only capable of generating a single type of association record, they are not associated with an observation point. An example of an association is a netblock record mapping a block address and a prefix with a block name, geographic location, and technical and administrative point-of-contact handles; the source of such an association might be a regional internet registry.

The inclusion of associations in the data model does not specifically support the translation of raw flow data from format to format, but it does support the translation of data into more aggregated or annotated formats, which may enrich the analysis products that can be derived from them.

Each association has three timestamps. The start and end timestamps define the time period during which the association is presumed to be valid; this can be used for applications such as historical DNS resolution or routing information. The third timestamp is the most recent update timestamp, which can be used to track the last time a given association was checked against its source database.

Future work will flesh out association subclasses; we hope to receive community feedback on the usefulness of the association mechanism and the types of associations presently used in aggregate analysis.

5 Candidate Translator Design

Given a data model through which to translate flows and other event data, a candidate design for a translator built around that model suggests itself. Each format will require both an input reader, to build objects in the common event data model from an input stream of a given input data format; and an output writer, to translate these common objects back into the desired output data format.

Each of these readers and writers could be written from scratch, handling their own disk I/O and other low level functions, but this is both relatively inflexible, and implies an inordinate duplication of effort. Instead, these readers and writers will be implemented in terms of I/O primitives that will handle low-level I/O. The flexibility gained by this approach could be applied to translate records from data sources other than regular files on disk, for example, from shared memory in a blackboard architecture, or directly off the wire in the case of IPFIX data

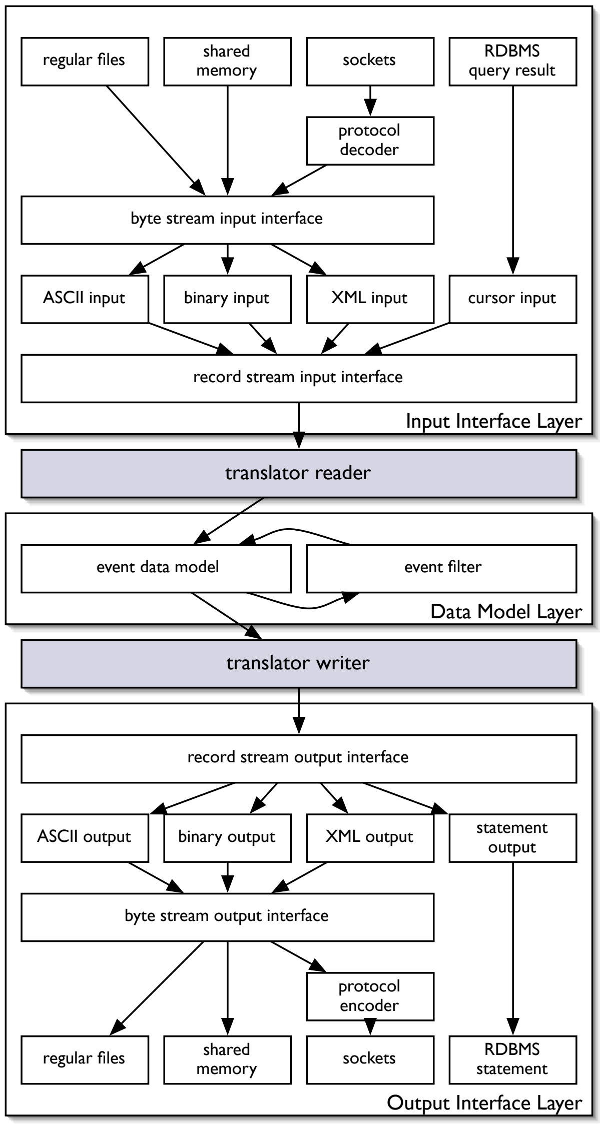


Figure 2: Candidate translator design data flow

collection.

In practice, there are limited number of ways which flow data can be represented in storage and transit, e.g. in fixed or variable length binary records, in delimited and separated ASCII records, as XML elements, as rows in a relational database. So the work of each format reader/writer author can be made less arduous still by providing an interface atop each of these fundamental formats.

Applying these principles, we are left with a data flow resembling Figure 2.

5.1 Event Model Implementation

We intend the data model to be implemented as a set of ANSI C structures, using structure containment to implement class inheritance. The choice of ANSI C was made for several reasons; most important are performance and portability.

The event data model is also designed to be implementable as a native storage format for the translator. Three implementations are planned: an XML Schema for text interchange of flow data, an IPFIX-compatible set of templated binary formats for binary interchange, and an RDBMS schema for long term storage and analysis in a relational database.

6 Conclusions and Future Directions

This paper has presented an event data model and a candidate translator design built around it, to facilitate the sharing of flow data among network security analysis communities. Core to the architecture of this translator is the realization that network security analysis tasks revolve around the manipulation of two classes of data, events and associations.

As of this writing, the ideas presented herein exist primarily on the whiteboard. The design pattern presented here has proven its usefulness in the cargogen tool released as part of AirCERT [3]; however, this tool is rather limited in its flexibility, shipping with only one input and output translator, and supporting only event data. Associations are supported after a fashion by the AirCERT AddrTree [4] RIR data source toolchain, although AddrTree does not support associations as a general superclass. These tools may be seen as ancestors of this present effort.

We plan to continue the design and implementation of this system during the summer and fall of 2005, focusing at first on flow translation, then on other types of events, finally implementing support for a variety of association subclasses.

References

[1] CLAISE, B. Ipfix protocol specification. Internet-Draft 15, Internet Engineering Task Force, May 2005. http://www.ietf.org/internet-drafts/draft-ietf-protocol-15.txt.

Notes

[1]http://www.qosient.com
[2]http://www.q1labs.com/products/prod_overview.html
[3]http://aircert.sourceforge.net
[4]http://aircert.sourceforge.net/addrtree

Working With Flow Data in an Academic Environment in the DDoSVax Project at ETH Zuerich

Arno Wagner

wagner@tik.ee.ethz.ch

Communication Systems Laboratory

Swiss Federal Institute of Technology Zurich (ETH Zurich)

Outline

1. Academic users
2. Context: The DDoSVax project
3. Data collection and processing infrastructure
4. Software / Tools
5. Technical lessons learned
6. Other lessons learned

Note: Also see my FloCon 2004 slides at `http://www.tik.ee.ethz.ch/~ddosvax/` or Google("ddosvax")

Academic Users

- PhD Researchers
- Students doing Semester-, (Diploma-) and Master-Theses
- (Almost) no forensic work

Users will write their own tools
$\Rightarrow$ Support is needed to make them productive fast:

- Software: Libraries, example tools, templates
- Initial explanations
- Advice and some supervision

The DDoSVax Project

`http://www.tik.ee.ethz.ch/~ddosvax/`

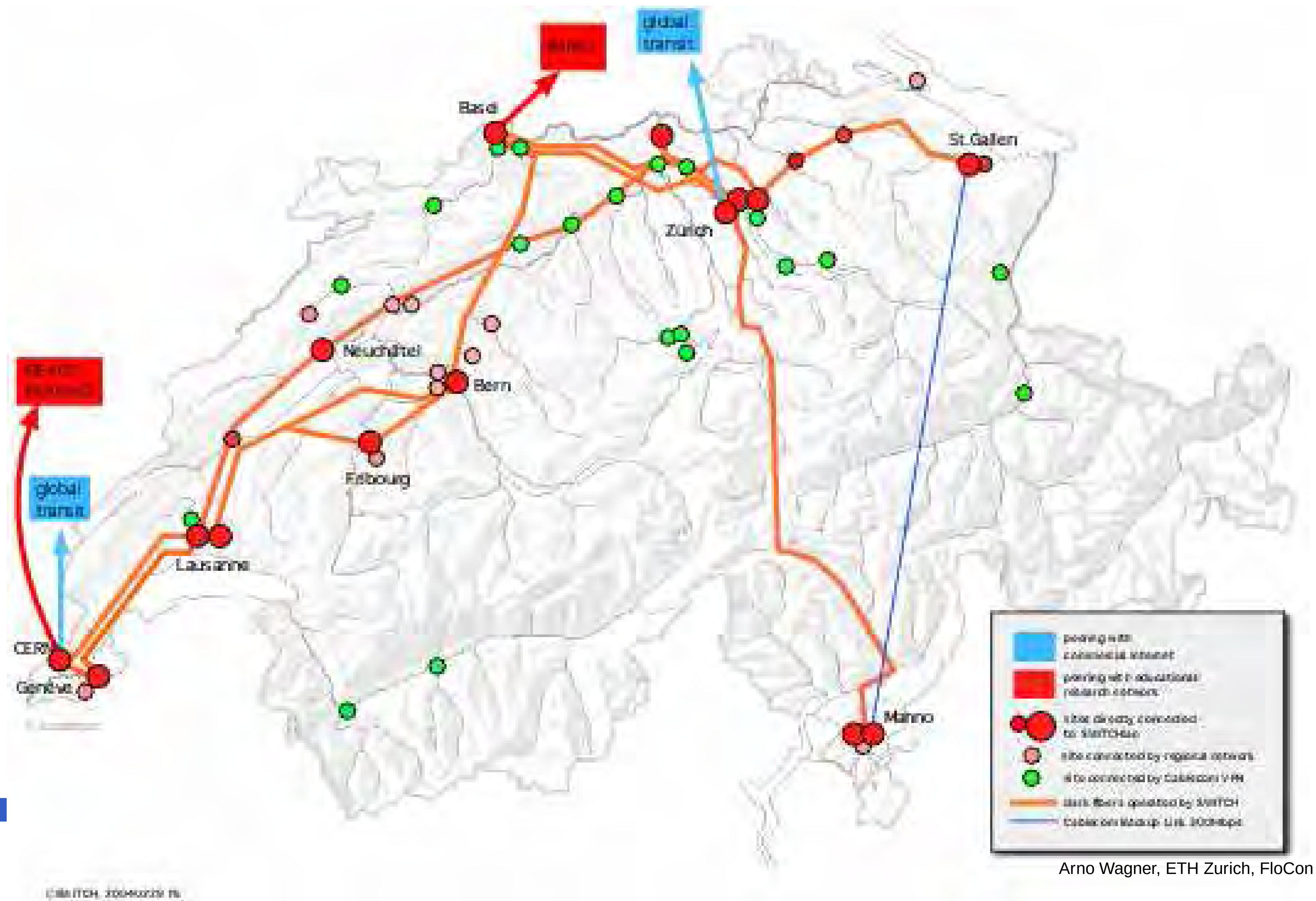
- Collaboration between SWITCH (www.switch.ch, AS559) and ETH Zurich (www.ethz.ch)
- Aim (long-term): Near real-time analysis and countermeasures for DDoS-Attacks and Internet Worms
- Start: Begin of 2003
- Funded by SWITCH and the Swiss National Science Foundation

DDoSVax Data Source: SWITCH

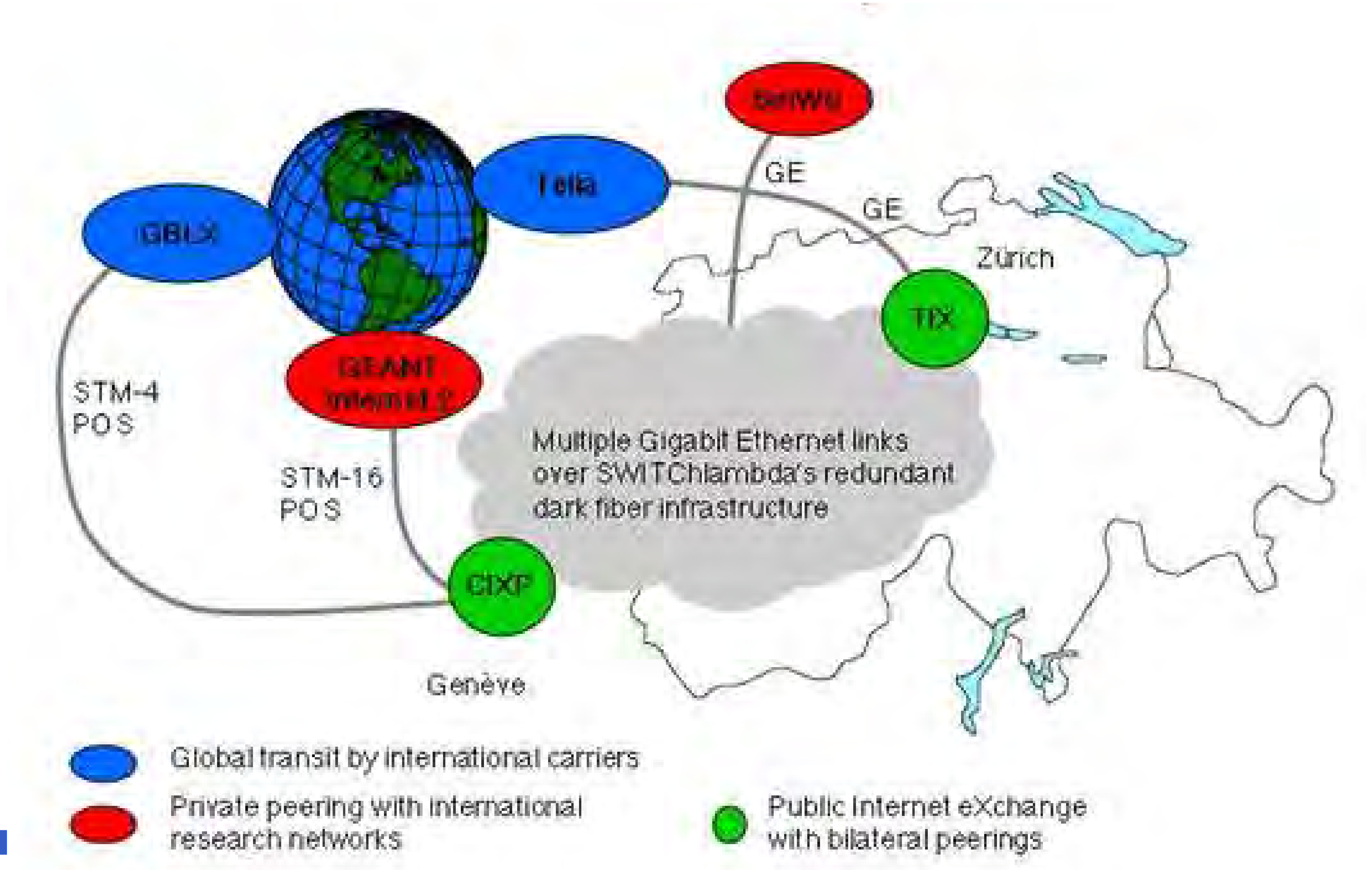
The Swiss Academic And Research Network

- .ch Registrar
- Links most Swiss Universities
- Connected to CERN
- Carried around 5% of all Swiss Internet traffic in 2003
- Around 60.000.000 flows/hour
- Around 300GB traffic/hour

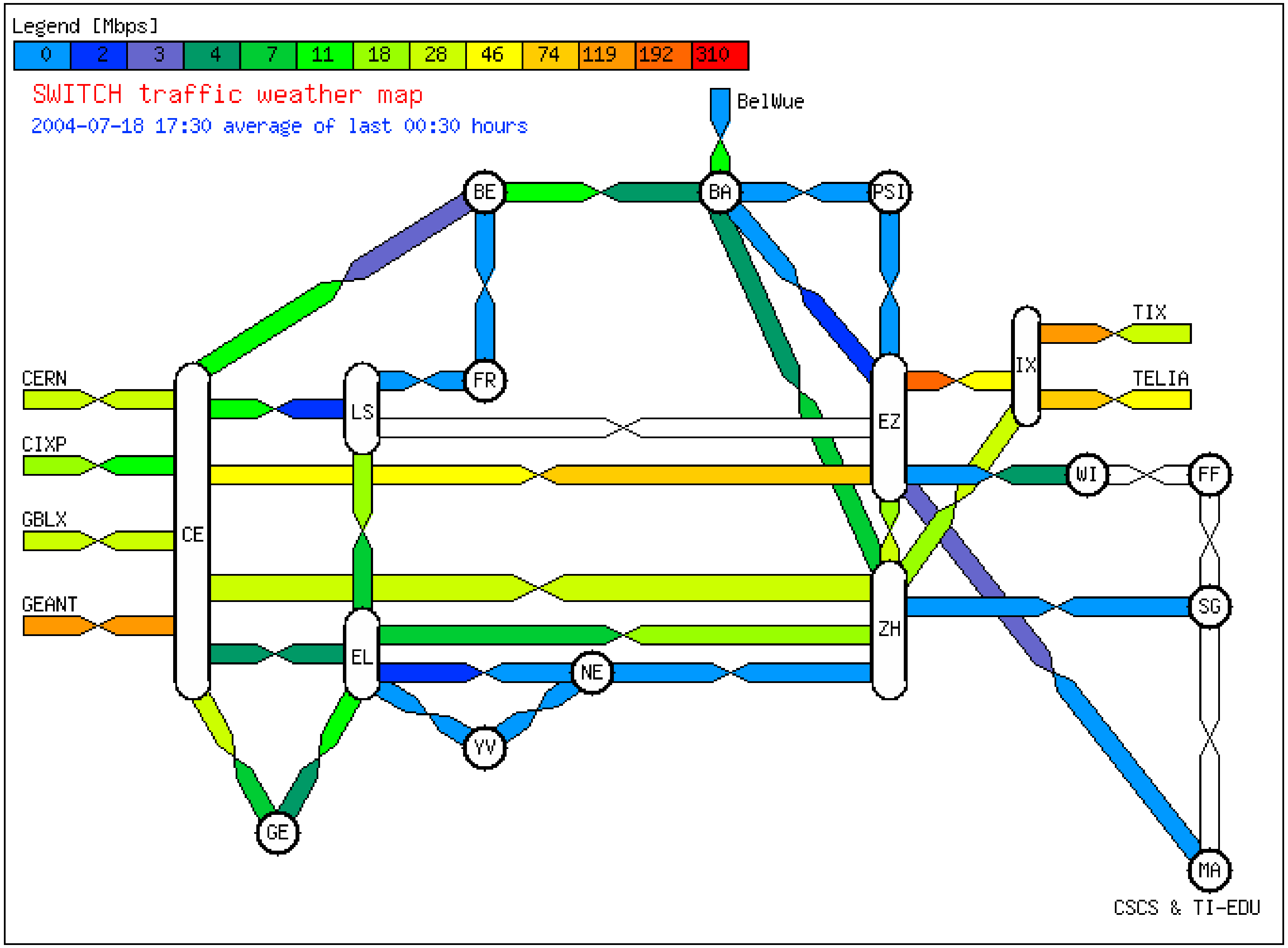
The SWITCH Network


global transit
Basel
St.Gallen
Zurich
Neuchâtel
Bern
Fribourg
global transit
Lausanne
CERN
Genève
Manno
peering with commercial internet
peering with educational research centers
sites directly connected to SWITCHlan
sites connected by regional network
sites connected by Cablecom VPN
dark fibers operated by SWITCH
Cablecom backup Link 300Mbps
©SWITCH

SWITCH Peerings


GBLX
Telia
GEANT
Internet 2
GE
GE
Zürich
TIX
STM-4
POS
STM-16
POS
Multiple Gigabit Ethernet links over SWITChlambda's redundant dark fiber infrastructure
CIXP
Genève
Global transit by international carriers
Private peering with international research networks
Public Internet eXchange with bilateral peerings

SWITCH Traffic Map



NetFlow Data Usage at SWITCH

- Accounting
- Network load monitoring
- SWITCH-CERT, forensics
- DDoSVax (with ETH Zurich)

Transport: Over the normal network

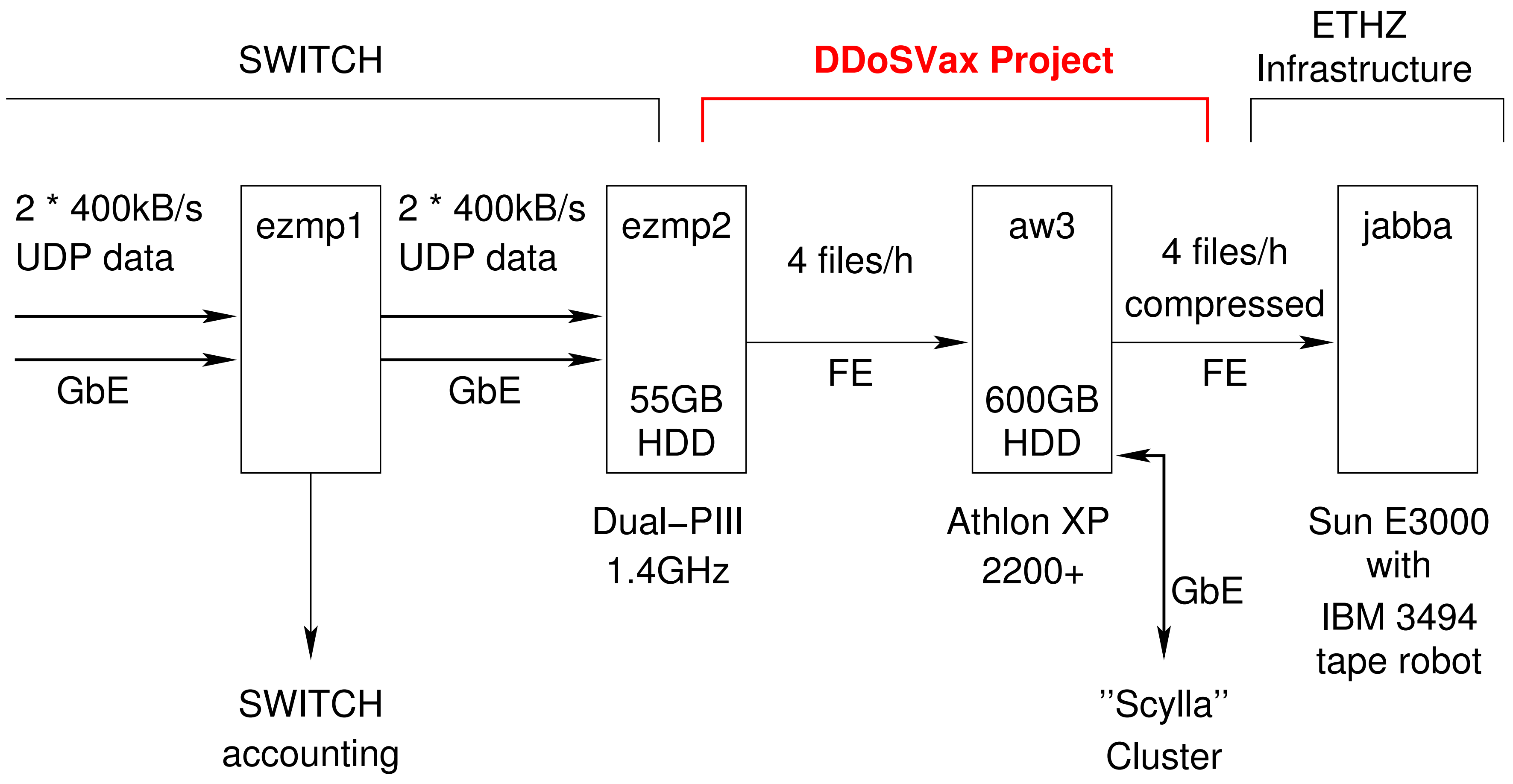
Collaboration Experience

- DDoSVax inspired SWITCH to crate their own short-term NetFlow archive for forensics
- Quite friendly and competent exchange with the (small, open minded) SWITCH technical and security staff.
- SWITCH may want to use our archive in the future as well
- Main issue with SWITCH: Privacy concerns

Network Dynamics

- No topological changes with regard to flow collection so far.
- Collection quality got better due to better hardware (routers).
- IP space (AS559) was a bit enlarged in the last year.

Collection Data Flow



NetFlow Capturing

- One Perl-script per stream
- Data in one hour files

Critical: (Linux) socket buffers:

- Default: 64kB/128kB max.
- Maximal possible: 16MB
- We use 2MB (app-configured)
- 32 bit Linux: May scale up to 5MB/s per stream

Capturing Redundancy

- Worker / Supervisor (both demons)
- Super-Supervisor (cron job)
  For restart on reboot or supervisor crash
- Space for 10-15 hours of data on collector

No hardware redundancy

Long-Term Storage

Unsampled flow-data since March 2003
Bzip2 compressed raw NetFlow V5 in one-hour files

- We need most data-fields and precise timestamps
- We don’t know what to throw away
- We have the archive space
- Causes us to be CPU bound (usually)
  $\Rightarrow$ Makes software writing a lot easier!

Computing Infrastructure

The ”Scylla” Cluster
Servers:

- aw3: Athlon XP 2200+, 600GB RAID5, GbE
  does flow compression and transfer
- aw4: Dual Athlon MP 2800+, 3TB RAID5, GbE
- aw5: Athlon XP 2800+, 400GB RAID5, GbE

Nodes:

- 22 * Athlon XP 2800+, 1GB RAM, 200GB HDD, GbE

Total cost (est.): 35 000 USD + 3 MM

Software

- Basic NetFlow libraries (parsing, time handling, transparent decompression, . . .)
- Small tools (conversion to text, statistics, packet flow replay, . . .)
- Iterator templates: Provide means to step through one or more raw data files one a record-by-record basis
- Support libraries: Containers, IP table, PRNG, etc.

All in c (gcc), commandline only. Most written by me. Partially specific to SWITCH data.

Lessons Learned (Technical)

Software:

- KISS is certainly valid.
- Unix-tool philosophy works well.
- Human-readable formats and Perl or Python are very useful for prototyping and understanding.
- Add information headers (commandline, etc.) to output formats (also binary)!
- Take care on monitoring the capturing system.
- Keep a measurement log!

Lessons Learned (Technical)

Hardware/OS:

- Needed much more processing power and disks storage than anticipated
  $\Rightarrow$ Plan for infrastructure growth!
- Get good quality hardware.

Lessons Learned (Technical)

Capturing and storage: Bit-errors do happen!
We use **bzip2 -1** on 1 hour files (about 3:1)

- Observed: 4 bit errors in compressed data/year
- 1 year ~ 5TB compressed $\Rightarrow 1$ error $/\ 1.2 * 10^{12}$ Bytes
- bzip2 -1 $\Rightarrow$ loss of about 100kB per error
  Unproblematic to cut defect part
  Note: gzip, lzop, ... will loose *all* data after the error
- Source of errors: RAM, busses, (CPU), (disk), (Network)

Lessons Learned (Technical)

Processing: Bit Errors do happen!

- Scylla-Cluster used OpenMosix $\Rightarrow$ Process migration and load balancing
- Observed problem: Frequent data corruption.
- Source: A single weak bit in 44 RAM modules
  Diag-time with memtest86: > 3 days!
  Process migration made it vastly more difficult to find!
- No problems with disks, CPUs, network, tapes.
- Some problems with a 66MHz PCI-X bus on a server.

Lessons Learned (Users)

Students need to understand what they are doing.

- Human-readable and scriptable output helps a lot!
- Clean sample code is essential.
- Tell students what technical skills are expected *clearly* before they commit to a thesis.
- Make sure students code cleanly and that they understand algorithmic aspects.

Thank You!

Flow-Data Compressibility Changes During Internet Worm Outbreaks

Arno Wagner Bernhard Plattner
Communication Systems Laboratory, Swiss Federal Institute of Technology Zurich
Gloriastr. 35, CH-8092 Zurich
Contact Author: Arno Wagner, wagner@tik.ee.ethz.ch

Abstract

During outbreaks of fast Internet worms the characteristics of network flow data from backbone networks changes. We have observed that in particular source and destination IP and port fields undergo compressibility changes, that are characteristic for the scanning strategy of the observed worm. In this paper we present measurements done on a medium sized Swiss Internet backbone (SWITCH, AS559) during the outbreak of the Blaster and Witty Internet worms and attempt to give a first explanation for the observed behaviour. We also discuss the impact of sampled versus full flow data and different compression algorithms. This is work in progress. In particular the details of what exactly causes the observed effects are still preliminary and under ongoing investigation.

1. Entropy and Compressibility

Generally speaking entropy is a measure of how random a data-set is. The more random it is, the more entropy it contains. Entropy contents of a (finite) sequence of values can be measured by representing the sequence in binary form and then using data compression on that sequence. The size of the compressed object corresponds to the entropy contents of the sequence. If the compression algorithm is perfect (in the mathematical sense), the measurement is exact.

On the theoretical side it is important to understand that not entropy is the relevant traffic characteristic, but Kolmogorov Complexity [16] of an interval of data. While entropy describes the average expected information content of a symbol that is chosen in a specific randomised way from a specific symbol set, Kolmogorov Complexity describes the specific information content of a specific object given, e.g. as a binary string of finite length.

2. Measurements

We are collecting NetFlow v5 [10] data from the SWITCH (Swiss Academic and Research Network [4], AS559) network, a medium-sized Swiss backbone operator, which connects all Swiss universities and various research labs (e.g. CERN) to the Internet. Unsampled NetFlow data from all four SWITCH border routers is captured and stored for research purposes in the context of the DDoSVax project [11] since early 2003. The SWITCH IP address range contains about 2.2 million IP addresses. In 2003 SWITCH carried around 5% of all Swiss Internet traffic [17]. In 2004, we captured on average 60 million NetFlow records per hour, which is the full, non-sampled number of flows seen by the SWITCH border routers.

In Figures 1,2,3 and 4 we plot the entropy estimations by compressibility over time for source and destination IP addresses and ports for the Blaster [9, 6, 15] and Witty [18, 20] worm. Both worms are relatively well understood and well documented. First observed on August 11th, 2003, Blaster uses a TCP random scanning strategy with fixed destination and variable source port to identify potential infection targets and is estimated to have infected 200'000...500'000 hosts worldwide in the initial outbreak. The Witty worm, first observed on March 20th, 2004, has some unexpected characteristics. Witty attacks a specific firewall product. It uses UDP random scans with *fixed* source port and *variable* destination port. Witty infected about 15'000 hosts in less than 20 minutes.

The y-axis in the plots gives inverse compression ration, i.e. lower values indicate better compressibility. The plotted time intervals start before the outbreaks to illustrate normal traffic compressibility characteristics. Samples taken from other times in 2003 and 2004 indicate that the pre-outbreak measurements, were source and destination figures are close together, are characteristic for non-outbreak situations. The outbreak times of both worms are marked with arrows.

The given measurements were done both on the full SWITCH flow set as well as on a 1 in 20 sample. Com-

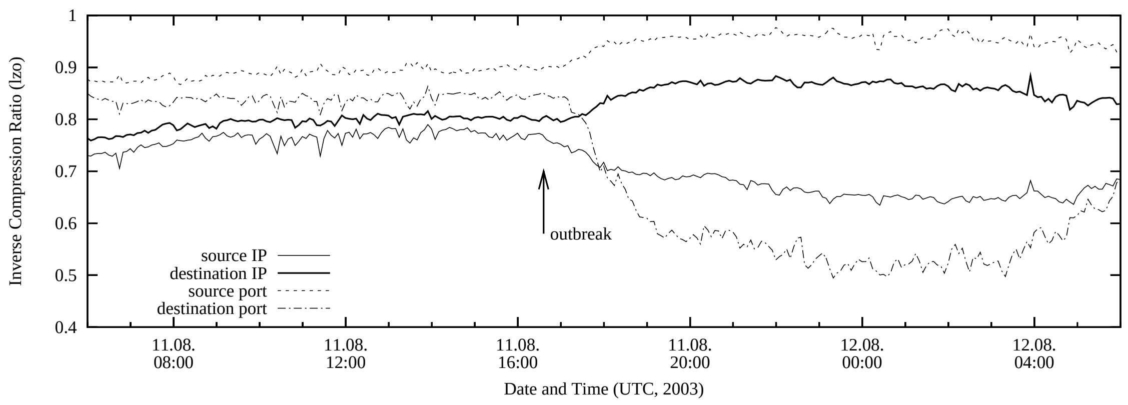


Figure 1. Blaster - TCP address parameter compressibility (lzo1x-1 algorithm)

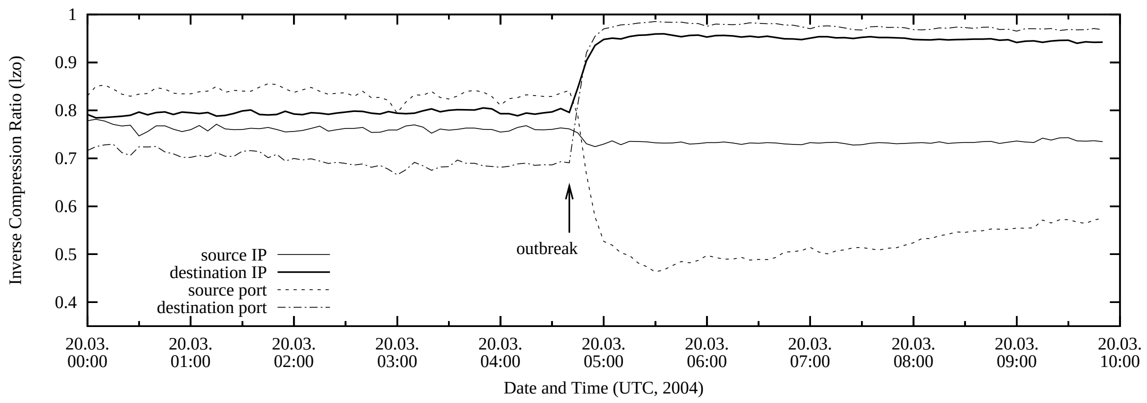


Figure 2. Witty - UDP address parameter compressibility (lzo1x-1 algorithm)

pression algorithm used is the fast lzo algorithm lzo1x-1 (see Section 4). It can be seen that in both cases the compressibility plots change significantly during the outbreak. Changes are consistent with the intuition that more random date is less compressible, while more structured date can be compressed better. The measurements on sampled data show a vertical shift, but still exhibit the same characteristic changes.

3. Analysis

In normal traffic there is roughly one return flow to a host for each flow it sends out as connection initiator. During a worm outbreak, most scanning flows do not have a return flow. This causes the changes in the overall flow data to be strongly dependent to the characteristics of the flows generated for scanning connection attempts. Note that the absence of an answering flow does not mean the absence of a host at the target address. It can also be due to firewalls, filters and not running services.

The connection between entropy and worm propagation is that worm scan-traffic is more uniform or structured than normal traffic in some respects and a more random in others. The change in IP address characteristics seen on a flow level is intuitive: few infected hosts try to connect to a lot of other hosts. If these flows grow to be a significant part of the set of flows seen in total, the source IP addresses of the scanning hosts will be seen in many flows and since they are relatively few hosts, the source IP address fields will contain less entropy per address seen than normal traffic. On the other hand the target IP addresses seen in flows will be much more random than in normal traffic. These are fundamental characteristics of any worm outbreak where each infected host tries to infect many others.

For ports, the behaviour is more variable. The typical scanning behaviour will be a random (from an OS selected range) or fixed source port and a fixed destination port. In the Blaster plots the impact of random source port and fixed destination port can be seen clearly. Witty is different. Because it did scan with fixed source port and random target port (because it attacked a firewall product that sees all network traffic), the port plots show exactly the opposite compressibility changes compared to Blaster.

At this time it in unclear how much weaker a topological worm (i.e. a worm that uses data from the local host to determine scanning targets and does not do random scanning) would influence the flow field compressibility statistics.

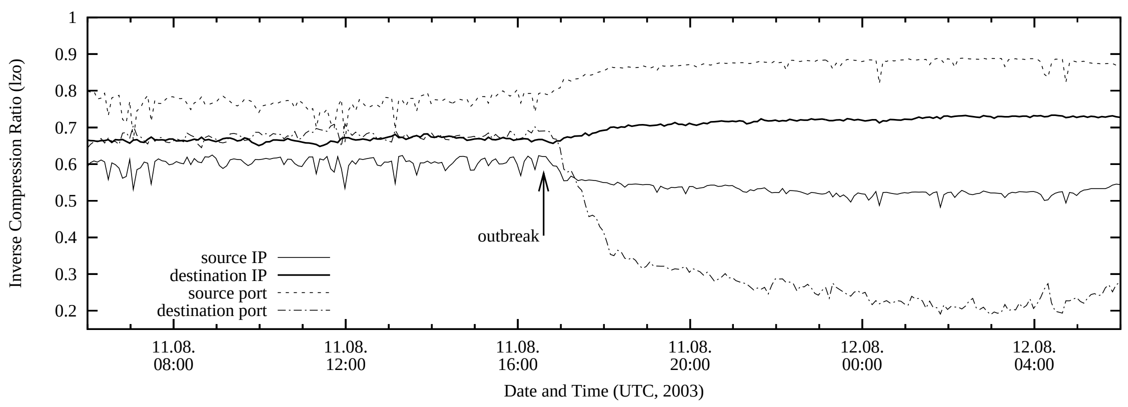


Figure 3. Blaster - TCP, randomly sampled at 1 in 20 flows (lzo1x-1 algorithm)

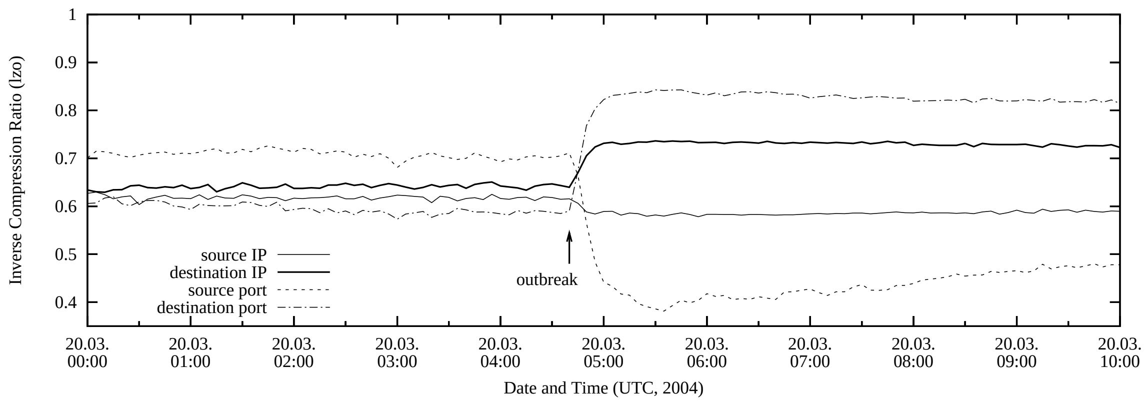


Figure 4. Witty - UDP, randomly sampled at 1 in 20 flows (lzo1x-1 algorithm)

4. Compressor Comparison

Method (Library)	CPU time / hour (60'000'000 flows/hour)
bzip2 (libbz2-1.0)	169 s
gzip (zlib1g 1.2.1.1-3)	52 s
lzo1x-1 (liblzo1 1.08-1)	7 s

Figure 6. CPU time (Linux, Athlon XP 2800+)

We compared three different lossless compression methods, the well-known bzip2 [2] and gzip [3] compressors as well as the lzo (Lempel-Ziv-Oberhumer) [1] real-time compressor. We did not consider lossy compressors. Bzip2 is slow and compresses very well, gzip is average in all regards and lzo family is fast but does not compress well.

Direct comparison of the three compressors on network data shows that while the compression ratios are different, the changes in compressibility are very similar. Figure 5 gives an example plot that compares the compression statistics for destination IP addresses before and during the Witty worm outbreak. Because of its speed advantage lzo1x-1 was selected as preferred algorithm for our work. Note that it is extremely fast (Table 6, non-overlapping measurement intervals of 5 minutes each, includes all overhead like NetFlow record parsing) and uses little memory (64kB for the compressor), making it far more efficient than other methods of entropy estimation, like for example methods based on determining the frequency of individual data values. Since we are only concerned with relative changes, the far from optimal compression ratio of the algorithm does not matter.

5. Related Work

The idea to use some entropy measurements to detect worms has been floating around the worm research community for some time. Yet we are not aware of any publication(s) describing concrete approaches, systems or measurements. The authors of this paper were prompted to investigate this idea by an observation on the Nachi [12, 7, 5] worm: Nachi generated about as many additional ICMP flow records as there were total flow records exported before the outbreak, yet the compressed size of the storage files increased only marginally.

In [19] the authors describe *behaviour-based clustering*, an approach that groups alerts from intrusion detection systems by looking at similarities in the observed packet header fields. The clusters are then prioritised for operator review. Principal Component Analysis is used in [14] to separate normal and attack

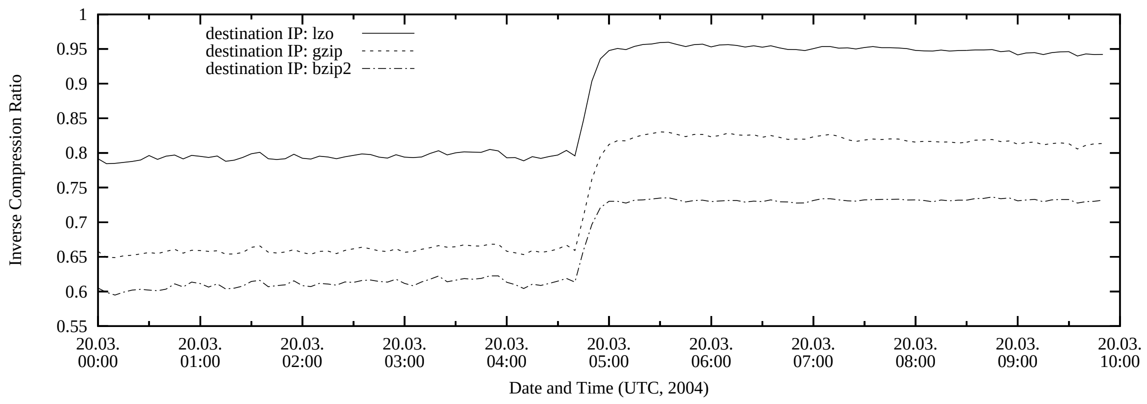


Figure 5. Witty - compressor comparison

traffic on a network-wide scale in a post-mortem fashion. Detection of exponential behaviour in a worm outbreak is studied in [8]. In [13] the authors study how worms propagate through the Internet.

6. Conclusion

We have presented measurements that indicate compressibility analysis of network flow data address fields can be used for the detection of fast worms. The approach is generic and does not need worm-specific parameterisation in order to be effective. It can generate first insights and is suitable for initial alarming, but has limited analytic capability. We are currently investigating how the entropy-based approach can help to generate a more detailed analysis of a massive network event.

References

[1] `http://www.oberhumer.com/opensource/lzo/`. LZO compression library.
[2] The bzip2 and libbzip2 official home page. `http://sources.redhat.com/bzip2/`.
[3] The gzip home page. `http://www.gzip.org/`.
[4] The swiss education & research network. `http://www.switch.ch`.
[5] Mcafee: W32/nachi.worm. `http://vil.nai.com/vil/content/v_100559.htm`, August 2003.
[6] Symantec Security Response - W32.Blaster.Worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.worm.html`, 2003.
[7] W32.welchia.worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.welchia.worm.html`, August 2003.
[8] C. C. Z. an L. Gao, W. Gong, and D. Towsley. Monitoring and Early Warning for Internet Worms. In *Proceedings of the 10th ACM Conference on Computer and Comminication Security*, 2003.
[9] CERT. Security Advisory: MS.Blaster (CA-2003-20). `http://www.cert.org/advisories/CA-2003-20.html`, 2004.
[10] Cisco. White Paper: NetFlow Services and Applications. `http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm`, 2002.
[11] DDoSVax. `http://www.tik.ee.ethz.ch/~ddosvax/`.
[12] H. Gabor Szappanos VirusBuster. Virus Bulletin: Virus information and overview - W32/Welchia. `http://www.virusbtn.com/resources/viruses/welchia.xml`, Apr. 2004.
[13] J. Kim, S. Radhakrishnan, and S. K. Dhall. Measurement and Analysis of Worm Propagation on Internet Network Topology. In *Proceedings of the International Conference on Computer Communications and Networks*, 2004.
[14] A. Lakhina, M. Crovella, and C. Diot. Diagnosing network-wide traffic anomalies. In *SIGCOMM*, pages 219–230, 2004.
[15] R. Lemos. MSBlast epidemic far larger than believed. `http://news.com.com/MSBlast+epidemic+far+larger+than+believed/2100-7349_3-5184439.html`, 2004.
[16] M. Li and P. Vitanyi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Verlag, second edition edition, 1997.
[17] O. Müller, D. Graf, A. Oppermann, and H. Weibel. Swiss Internet Analysis. `http://www.swiss-internet-analysis.org/`, 2004.
[18] C. Shannon and D. Moore. CAIDA: The Spread of the Witty Worm. `http://www.caida.org/analysis/security/witty/`, 2004.
[19] K. Theriault, D. Vukelich, W. Farell, D. Kong, and J. Lowry. Network traffic analysis unsing behaviour-based clustering. Whitepaper, BBN Technologies, `http://www.bbn.com/docs/whitepapers/NetTrafficAn-Clustering-Theriault10-02.pdf`.
[20] US-CERT. Vulnerability Note: Witty (VU#947254). `http://www.kb.cert.org/vuls/id/947254`, 2004.

Identifying P2P Heavy-Hitters from Network-Flow Data

Arno Wagner* Thomas Dübendorfer* Lukas Hämmerle† Bernhard Plattner*
*Communication Systems Laboratory, Swiss Federal Institute of Technology Zurich,
Gloriastr. 35, CH-8092 Zurich, {wagner, duebendorfer, plattner}@tik.ee.ethz.ch
† SWITCH, PO Box, CH-8021 Zurich, Switzerland, haemmerle@switch.ch

Contact Author: Arno Wagner, phone: +41 44 632 7004, fax: +41 44 632 10 35

Abstract

One major new and often not welcome source of Internet traffic is P2P filesharing traffic. Banning P2P usage is not always possible or enforcible, especially in a university environment. A more restrained approach allows P2P usage, but limits the available bandwidth. This approach fails when users start to use non-default ports for the client software. The PeerTracker algorithm, presented in this paper, allows detection of running P2P clients from NetFlow data in near real-time. The algorithm is especially suitable to identify clients that generate large amounts of traffic. A prototype system based on the PeerTracker algorithm is currently used by the network operations staff at the Swiss Federal Institute of Technology Zurich. We present measurements done on a medium sized Internet backbone and discuss accuracy issues, as well as possibilities and results from validation of the detection algorithm by direct polling in real-time.

1. Introduction

P2P filesharing generates large amounts of traffic. It seems even to be one of the driving factors for home-users to get broadband Internet connections. It also has become a significant factor in the total Internet bandwidth usage by universities and other organisations. While in some environments a complete ban on P2P filesharing can be a solution, this gets more and more difficult as legitimate uses grow. The Swiss Federal Institute of Technology at Zurich (ETH Zurich) has adopted an approach of allowing P2P filesharing, but with limited bandwidth. The default ports of the most popular P2P filesharing applications are shaped to a combined maximum Bandwidth of 10Mbit/s. There is a relatively small number of ”heavy hitters” that consume a large share of the overall P2P bandwidth and avoid the use of default ports and hence the bandwidth limitations. In fast network connections, such as the gigabit ETH Internet connectivity, it is difficult to identify and monitor P2P users for their bandwidth consumption. If heavy hitters can be identified, they can be warned to reduce their bandwidth usage or, if that does prove ineffective, special filters or administrative action can be used against them. In this way P2P traffic can be reduced without having to impose drastic restrictions on a larger user population.

To this end, we have developed the *PeerTracker* algorithm that identifies P2P users based on Cisco NetFlow [4]. It determines hosts participating in the most common P2P networks and detects which port setting they use. This information can then be used to determine P2P bandwidth usage by the identified hosts. We present the PeerTracker algorithm as well as results from measurements done in the SWITCH [3] network, a medium sized Internet backbone in Switzerland. We discuss detection accuracy issues and give the results of work done on validation of the PeerTracker algorithm by real-time polling of identified P2P hosts. Note that the PeerTracker cannot identify which files are actually shared, since it only sees flow data. The PeerTracker can track currently track clients for the eDonkey, Overnet, Kademlia (eMule), Gnutella, FastTrack, and BitTorrent P2P networks.

A prototypical implementation of the PeerTracker algorithm, fitted with a web-interface, is currently in use at the central network services of ETH Zurich in a monitoring-only set-up for hosts in the ETH Zurich network. A software release under the GPL is planned.

2. DDoSVax project

The DDoSVax[5] project maintains a large archive NetFlow[4] data which is provided by the four border gateway routers of the medium-sized backbone AS559 network operated by SWITCH[3]. This network connects all Swiss universities, universities of applied sciences and some research institutes. The SWITCH IP address range contains about 2.2 million addresses, which approximately corresponds to a /11 network. In 2003, SWITCH carried around 5% of all Swiss Internet traffic [9]. In 2004, on average 60 million NetFlow records per hour were captured, which is the full, non-sampled number of flows seen by the SWITCH border routers. The data repository contains the SWITCH traffic data starting from the beginning of 2003 to the present.

3. PeerTracker: Algorithm

P2P traffic can be TCP or UDP. We use the term "default port of a P2P system" to also include the choice of TCP or UDP.

Figure 1 shows the PeerTracker state diagram for each individual host seen in the network. When a network connection is detected each endpoint host becomes a *candidate peer*. A *candidate peer* that has additional P2P traffic becomes an *active peer* and is reported as active. Otherwise is becomes a *non-peer* after it has had no P2P traffic for a probation period (900 seconds) and is deleted. Each *active peer* is monitored for further P2P activity. After a maximum time without P2P traffic (600 seconds) it becomes a *dead peer*. Each *dead peer* is still monitored for P2P activity but not reported as active anymore. When a *dead peer* has P2P activity, it becomes active again. After a second time interval, the maximum afterlife (1 hour) without P2P activity a *dead peer* is considered gone and is deleted from the internal state of the PeerTracker.

The decision whether a specific network flow is a P2P flow is made based on port information. If a P2P client uses a non-default listening port (e.g. in order to circumvent traffic shaping) the peer still will communicate with other peers on using the default port(s) from time to time. The last 100 local and remote ports (TCP and UDP) are stored for every observed host, together with the amount of traffic on the individual ports. Traffic with one or both ports not in the range 1024-30000 (TCP and UDP) is ignored, since we found that most P2P traffic uses these ports. With reasonable threshold values on traffic amount (different for host within the SWITCH network and hosts outside) the most used local and remote ports allow the determination which P2P network a specific host participates in. This is done at the end of every measurement interval (900 seconds). Although some hosts can be part of several P2P networks only the one they exchange the most date with is identified.

We determine a lower and an upper bound for the total amount of P2P traffic. The lower bound is all P2P traffic were at least one side uses a default port. The upper bound also counts all traffic were source and destination ports are above 1023 and one side was identified as P2P host. The effective P2P traffic is expected to be between these two bounds, and likely closer to the upper bound, because in particular P2P heavy-hitters rarely run other applications that cause large amounts of traffic with port numbers above 1023 on both sides. Typical non-P2P applications with port numbers on both sides larger than 1023 are audio and video streaming and online gaming, all of which do not run well on hosts that also run a P2P client.

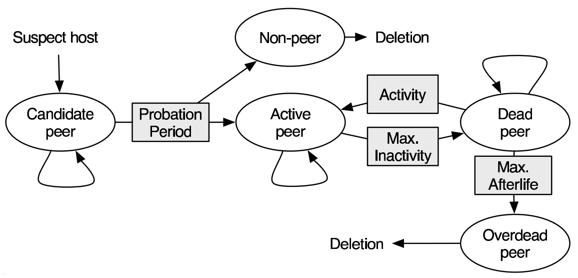


Figure 1. PeerTracker hosts state diagram

4. PeerTracker: Measurements

Due to traffic encryption and traffic hiding techniques used by some current P2P systems, the accurate identification of P2P traffic is difficult, even if packet inspection methods are used. Nevertheless, our NetFlow based approach can provide good estimations for the effective P2P traffic, even for networks with gigabit links that could hardly be analysed with packet inspection methods.

Identification of peers and their traffic is especially difficult if they have a low activity. This is an issue for all two-tier systems in which ordinary peers mainly communicate with a super peer and have few file transfers. Peers from one-tier systems like Overnet can be identified better because they communicate with many other peers even if no file transfers are in progress.

P2P traffic in the SWITCH network is quite substantial. The lower bound for P2P traffic (stateless P2P default port identification) significantly lower than the upper bound for all observed P2P systems (Table 2), which means that quite some P2P traffic cannot

P2P System	Default port usage
BitTorrent	70.0 %
FastTrack	8.3 %
Gnutella	58.6 %
eDonkey	55.6 %
Overnet	93.9 %
Kademlia	66.6 %

Table 1. P2P ports, SWITCH network, August 2004

be accurately estimated using only a stateless P2P default port method. The upper bound P2P traffic was about 24% (holiday, August 2004), 27% (non-holiday) respectively, of the total traffic that passed through the SWITCH border routers.

BitTorrent P2P users cause about as much traffic as eDonkey, Overnet and Kademlia users together, as can be seen in Figure 2. All peers of the SWITCH network generate 1.6 times more traffic to non-SWITCH hosts than incoming traffic, thus making the SWITCH network a content provider. This is probably due to the fast Internet connection most SWITCH users have and the traffic shaping mechanisms that some universities in the SWITCH network use. Users within the university network hope to evade the traffic limiting by using non-default listening ports.

5. Result Validation

The PeerTracker tries to identify P2P hosts and the used P2P network only on network flows seen, but makes no attempt to check its results in any other way. It is completely invisible on the network. There are two possible failure modes: *False positives* are hosts that the PeerTracker reports as having a P2P client running, while in fact they do not. *False negatives* are hosts that run a P2P client but are not identified by the PeerTracker. It is difficult to identify false negatives. From manual examination of the flow-level data and comparison with the PeerTracker output we found that while there are unidentified P2P clients, these hosts have only very limited P2P activity and do not contribute significantly to the overall traffic. This is consistent with the intuition that the PeerTracker algorithm can identify hosts with a lot of P2P much more easily than those with little traffic.

In order to identify false negatives, we have implemented an experimental extension to the PeerTracker that tries to determine whether hosts identified by the PeerTracker are actually running the indicated P2P client by actively polling them over the network.

P2P System	TCP	P2P-client
eDonkey, Overnet, Kademlia	50%	41%
Gnutella	53%	30%
FastTrack	51%	41%
Total	51%	38%

Table 4. Positive polling answers

Polling for all networks was done with TCP only. Table 3 gives a short overview of the polling methods used.

The results of a representative measurement from February 2005 can be found in Table 4. It can be seen that roughly half of the identified hosts are not reachable via TCP at all, likely due to Network Address Translation (NAT) and firewalls that prevent connections initiated by outside hosts. Assuming that reachable and unreachable hosts have similar characteristics with regard to their P2P traffic, the the difference between TCP-reachable hosts and positive polling results presents an upper limit for the number of false positives. The reasons for unsuccessful P2P client polling identified in a manual analysis are that the PeerTracker sometimes reports the wrong P2P network for a host, that especially Gnutella hosts answer in a variety of ways, some not expected by the polling code, and misdetection by the PeerTracker algorithm.

6. Related Work

While there are numerous measurements studies that use packet inspection [13, 7, 12, 8] for traffic identification, recently some have been published that use flow-level heuristics. In [14] signalling and download traffic was measured in a large ISP network using stateless default port number detection. Considered P2P networks were FastTrack, Gnutella and Direct Connect. An interesting approach is presented in [10]. The idea is to relate flows to each other according to source and destination port numbers using a flow relation map heuristic with priorities and SYN/ACKs to identify listening port. In [15] packet headers (first 64 bytes) from a campus network and the network of a research institute with about 2200 students and researchers were used as basis of P2P measurements. Flow measurements in the backbone of a large ISP were done in [6] for May 2002 and January 2003. The researchers determined the server port using the IANA [2] port numbers and the more detailed Graffiti [1] port table, giving precedence to well-known ports. Unclassified traffic was grouped in a ”TCP-big” class that includes flows with more than 100 KB data transmitted in less than 30 minutes.

P2P System	P2P lower bound	P2P upper bound
BitTorrent	55.4 Mbit/s (12.2 %)	90.1 Mbit/s (19.9 %)
FastTrack	1.8 Mbit/s (0.4 %)	12.3 Mbit/s (2.7 %)
Gnutella	5.1 Mbit/s (1.1 %)	10.7 Mbit/s (2.4 %)
eDonkey, Overnet, Kademlia	47.7 Mbit/s (10.5 %)	82.1 Mbit/s (18.1 %)
Total P2P	110.0 Mbit/s (24.4 %)	195.2 Mbit/s (43.1 %)

Table 2. P2P traffic bounds and percentage of total SWITCH traffic (August 2004)

P2P System	Polling method	
FastTrack	Request:	`GET /.files HTTP/1.0`
	Response:	`HTTP 1.0 403 Forbidden` <number 1> <number 2>
	or	`HTTP/1.0 404 Not Found/nX-Kazaa-`<username>
Gnutella	Request:	`GNUTELLA CONNECT/`<version>
	Response:	`Gnutella` <status>
eDonkey, Overnet, Kademlia	Request:	Binary: 0xE3 <length> 0x01 0x10 <MD4 hash> <ID> <port>
	Response:	Binary: 0xE3 . . .
eMule	Same as eDonkey, but replace initial byte with 0xC5.	
BitTorrent	Unsolved. Seems to need knowledge of a shared file on the target peer.	

Table 3. Polling methods for different P2P clients (TCP, to configured port)

7. Conclusions

We presented an efficient P2P client detection, classification and population tracking algorithm that uses flow-level traffic information exported by Internet routers. It is well suited to find and track heavy-hitters of the eDonkey, Overnet, Kademlia (eMule), Gnutella, FastTrack, and BitTorrent P2P networks. We also validated detected peers by an application-level polling. Our results confirmed a good lower accuracy bound that is well suited for P2P heavy hitter detection. However, it is not optimally suited to detect low traffic P2P nodes. A validation of BitTorrent clients was not possible due to the specifics of this network. In addition we stated measurement results obtained with the PeerTracker and observations made during the validation efforts.

References

[1] Graffiti. `http://www.graffiti.com/services/` (July 2004).

[2] IANA. `http://www.iana.com/assignments/port-numbers/services/` (July 2004).

[3] The Swiss Education & Research Network. `http://www.switch.ch`.

[4] Cisco. White Paper: NetFlow Services and Applications. `http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm`, 2002.

[5] DDoSVax. `http://www.tik.ee.ethz.ch/~ddosvax/`.

[6] A. Gerber, J. Houle, H. Nguyen, M. Roughan, and S. Sen. P2P, The Gorilla in the Cable. Technical report, AT&T Labs - Research, June 2004.

[7] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. Technical report, October 2003.

[8] T. Karagiannis, A. Broido, and M. Faloutsos. File-sharing in the Internet: A characterization of P2P traffic in the backbone. Technical report, University of California, Riverside Department of Computer Science, November 2003.

[9] O. Müller, D. Graf, A. Oppermann, and H. Weibel. Swiss Internet Analysis. `http://www.swiss-internet-analysis.org/`, 2003.

[10] J.-J. K. Myung-Sup Kim and J. W. Hong. Towards Peer-to-Peer Traffic Analysis. Technical report, POSTECH, Korea, October 2003.

[11] M. Ripeanu. Peer-to-Peer Architecture Case Study: Gnutella Network. In *Proceedings of the First International Conference on Peer-to-Peer Computing (P2P01)*. IEEE, 2001.

[12] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy. Analysis of Internet Content Delivery Systems. Technical report, University of Washington, December 2002.

[13] S. Sen and J. Wang. Accurate, Scalable In-Network Identification of P2P Traffic Using Application Signatures. Technical report.

[14] S. Sen and J. Wang. Analyzing peer-to-peer traffic across large networks. Technical report, AT&T Labs - Research, November 2002.

[15] R. van de Meent and A. Pras. Assessing Unknown Network Traffic. Technical report, University of Twente, Enschede, October 2003.

VisFlowConnect-IP: An Animated Link Analysis Tool For Visualizing Netflows [*]

Xiaoxin Yin William Yurcik Adam Slagell
National Center for Supercomputing Applications (NCSA)
University of Illinois at Urbana-Champaign
{xiaoxin,byurcik,slagell}@ncsa.uiuc.edu

Abstract

We present VisFlowConnect-IP, a network flow visualization tool that allows operators to detect and investigate anomalous internal and external network traffic. We model the network on a parallel axes graph with hosts as nodes and traffic flows as lines connecting these nodes. We present an overview of this tool's purpose, as well as a detailed description of its functions.

1 Introduction

Networks are becoming increasingly complex, and the number of different applications running over them is growing proportionally. No longer can a system/network administrator realisitically be aware of every application on every machine under her control. At the same time, the number of network attacks against machines has increased exponentially. These attacks are often concealed among this vast amount of legitimate, and seemingly random, traffic. It is often difficult just to log this traffic, yet alone analyze and detect attacks in real-time with traditional text-based tools.

However, humans excel at processing visual data and identifying abnormal patterns. Visualization tools can translate the myriads of network logs into animations that capture the patterns of network traffic in a succinct way, thus enabling users to quickly identify abnormal patterns that warrant closer examination. Such tools enable network administrators to sift through gigabytes of daily network traffic more effectively than scouring text-based logs.

VisFlowConnect-IP is one such network visualization tool. It visualizes network traffic as a parallel axes graph with hosts as nodes and traffic flows as lines connecting these nodes. These graphs can then be animated over time to reveal trends. VisFlowConnect-IP has the following distinguishing features: (1) it uses animations to visualize network traffic, so that network dynamics can be presented to users in a comprehensible and efficient manner, (2) it provides both an overview of traffic as well as drill-down views that allow users to dig out detailed information, and (3) it provides filtering capabilities that enables users to remove mundane traffic details from the visualization.

2 System Architecture

The general system architecture of VisFlowConnect-IP is shown in Figure 1. VisFlowConnect-IP has three main components: (1) an agent that extracts NetFlow records, (2) a NetFlow analyzer that processes the raw data and stores important statistics, and (3) a visualizer that converts the statistics into animations. In this section, we describe the design and implementation of each of the 3 components.

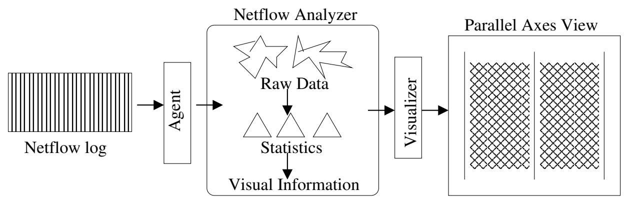


Figure 1. System Overview

2.1 NetFlow Source Data

VisFlowConnect-IP can use the following NetFlow formats: Cisco 5/7 and Arugs[1]. VisFlowConnect-IP works in a batch mode, reading NetFlow records from a log. An agent is used to extract the NetFlow records and feed them into VisFlowConnect-IP. Each record contains the following information: (1) sourcce/destination IP addresses and ports, (2) number of bytes and packets, (3) start and end timestamps, and (4) protocol type.

2.2 Input Filtering Capability

NetFlow logs contain many different types of traffic with distinct properties. While certain traffic patterns are usually a red flag, depending upon the context, they may be quite normal and benign. For example, it is very common that a DNS server has connections with every other

[*]This research was supported in part by a grant from the Office of Naval Research (ONR) under the auspices of the National Center for Advanced Secure Systems Research (NCASSR) <http://www.ncassr.org>

[1]http://www.qosient.com/argus/

host on a network, but on a workstation this may indicate a worm infection. In order to remove noise such as this, VisFlowConnect-IP provides advanced filtering profiles that users can store and load.

Let $F_1, \ldots, F_k$ be a set of user created filters. Table 1 shows filter variables and their value ranges. Each filter has a list of constraints on the variables and a leading label ("+" or "–") that indicates whether to "include" or "exclude" matches. A constraint on a variable takes the form of "$x = v_{min} - v_{max}$", where "x" is a variable and "v_{min}" and "v_{max}" are the lower and upper bounds of "x", and "$=$" is the only operator defined. Records are passed sequentially through each filter and the last match will determine whether or not to include the record. A record that matches no filter rules is dropped. For example, the following set of filters will include all traffic from domain 141.142.x.x with a source port between 1 and 1000, except tcp traffic involving port 80.

+: (SrcIP=141.142.0.0-141.142.255.255), (SrcPort=1-1000)

-: (SrcPort=80, Protocol=tcp)

-: (DstPort=80, Protocol=tcp)

Variables	Value Ranges
SrcIP, DstIP	0.0.0.0 ⇔ 255.255.255.255
SrcPort, DstPort	0 ⇔ 65535
Protocol	tcp, udp, icmp
PacketSize	0 ⇔ ∞

Table 1. Input Filter Language

3 How to Use VisFlowConnect-IP

In this section, we describe the visualize interface of VisFlowConnect-IP—which can be downloaded at <http://security.ncsa.uiuc.edu/distribution/VisFlowConnectDownLoad.html>

In *the Parallel Axes View*, three vertical axes are used to indicate traffic between external domains and internal hosts on the center axis (Figure 3). Points on the left [right] axis represent external domains that are sourcing [receiving] flows to [from] the internal network. Unlike the middle axis where points represent individual hosts, here points represents sets of hosts. The darkness of a line between two points is proportional to the logarithm of traffic volume between the hosts. All points are sorted according to their IP addresses, so that each point will remain at a relatively stable position for a user to track during animation. Figure 2 illustrates the VisFlowConnect-IP GUI with important features labeled.

1. **Menu Bar:** It contains the menu items for operations that are less frequently used, including (1) 'Open': open a NetFlow file, (2)'Load Filters': load a file for input filters, (3) 'Settings': bring up the settings dialog box, (4) 'Show Domain': show the domain view of the selected domain (described below), (5) 'Host Statistics': show the traffic statistics of the selected host/domain, and (6) 'Save Screen': save a snapshot of the current view.

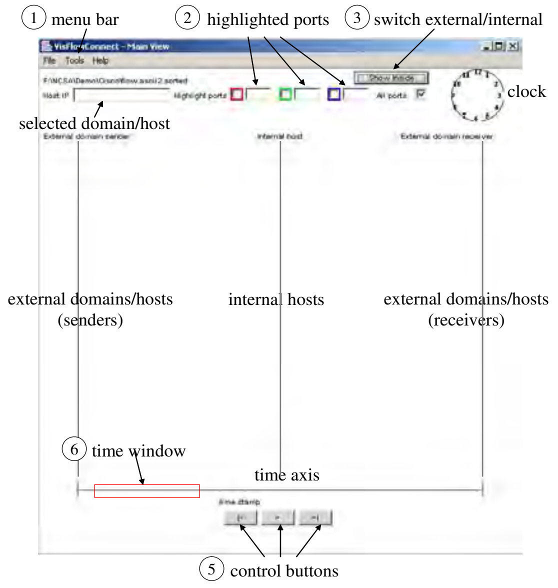


Figure 2. Parallel Axes View

2. **Highlighted Ports:** The user may specify up to three ports to highlight in special colors: red, green, or blue (see Figure 3). The user may also click on the check box to show traffic only on the highlighted port.
3. **External/Internal Switch:** The internal view (See figure 4) shows traffic between hosts on the internal network. The points on the left [right] axis represent the source [destination] of traffic flows. The user may switch between external and internal views by clicking on the button "Show Inside/Outside".
4. **Domain View:** As shown in Figure 5, VisFlowConnect-IP has a drill-down Domain View that allows a user to visualize traffic between hosts in a specific external network domain to/from hosts in the internal network. The Domain View shows all traffic between individual hosts in the corresponding external network domain and the internal network.
5. **Control Buttons:** A user can control the animation with three buttons: (| <) rewind back to start, (>) play forward a defined time unit (default is 10 minutes), and (> |) play forward to the end of the data set.
6. **Time Window:** Because a user will typically be more interested in recent traffic, only flows within a specified time window are shown as opposed to a cumulative view. A sliding rectangle along a horizontal time axis is shown at the bottom of the GUI to indicate the time window in view.
7. **Settings Dialog:** Figure 6 shows the settings dialog, which allows the user to change the input file format

(Cisco or Argus) and to select protocols of interest (e.g., tcp, udp or icmp). Here, the user may also adjust the traffic threshold, so that only domains whose aggreagate traffic volume is lower than this threshold are ignored. It also allows the user to change the time window, to restrict investigation to flows whose sizes are within a user-defined range, and to ignore flows over certain ports. This is also where the user sets the "Local IP Range" in order to distinguish internal hosts from external hosts.

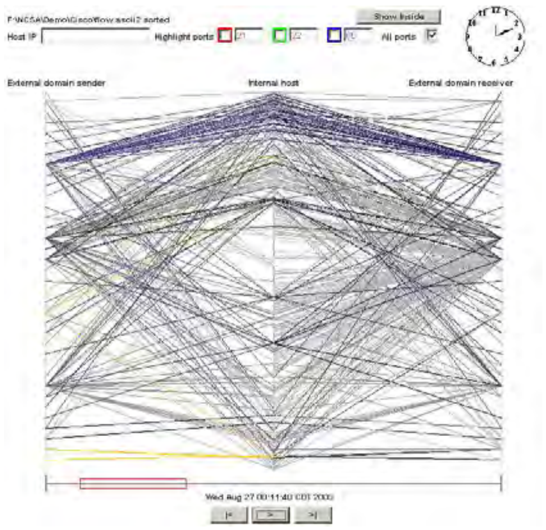


Figure 3. External View

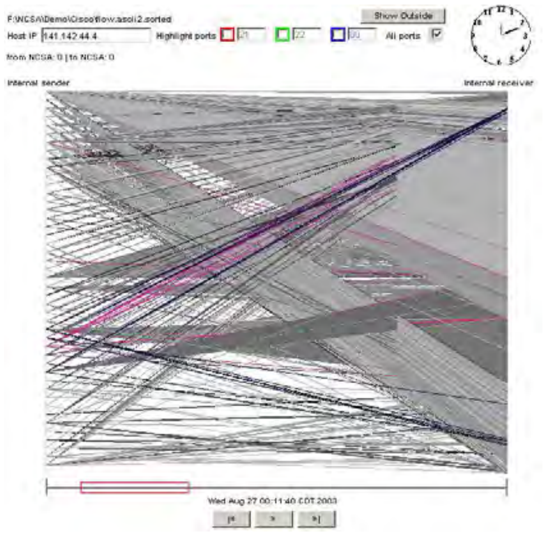

F:\NCSA\Demo\Ciscoflow.ascii2.sorted
Show Outside
Host IP 141.142.44.4
Highlight ports
All ports
from NCSA: 0 | to NCSA: 0
Internal sender
Internal receiver
Wed Aug 27 00:11:40 CDT 2003

Figure 4. Internal View

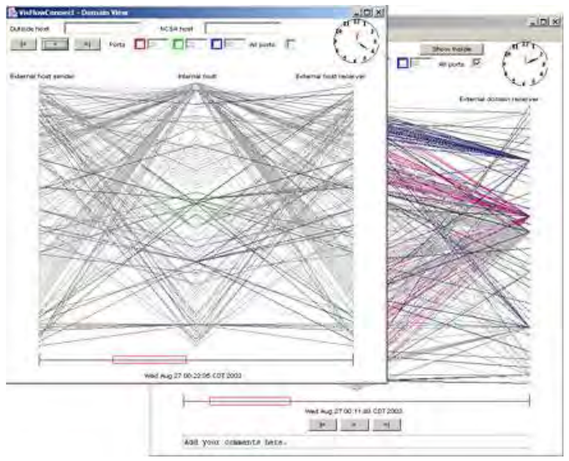
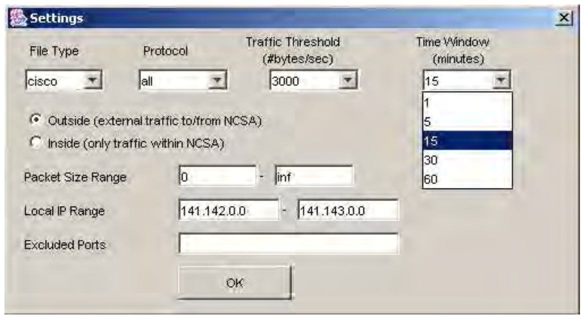


Figure 5. Domain View



Settings

File Type: cisco
Protocol: all
Traffic Threshold (#bytes/sec): 3000
Time Window (minutes): 15
1
5
15
30
60

(● Outside (external traffic to/from NCSA)
○ Inside (only traffic within NCSA)

Packet Size Range: 0 - inf
Local IP Range: 141.142.0.0 - 141.143.0.0
Excluded Ports:

OK

Figure 6. Setting Dialog

4 Related Work

In [5] the authors present a tool named NVT (Network Vulnerability Tool) that visually depicts a network topology and generates a vulnerability database. In [6], the authors present a visualization of network routing information that can be used to detect inter-domain routing attacks and routing misconfigurations. In [7], they go further and propose different ways of visualizing routing data in order to detect intrusions. An approach for comprehensively visualizing computer network security is presented in [4], where Erbacher et al. visualize the overall behavioral characteristics of users for intrusion detection. [1] focuses on visualizing log data from a web server in order to identify find patterns of malicious activity caused by worms.

Linkages among different hosts and events in a computer network contain important information for traffic analysis and intrusion detection. Approaches for link analysis are proposed in [2, 3, 8]. [2] and [8] focus on visualizing linkages in a network, and [3] focuses on detecting attacks based on fingerprints. Link analysis can illustrate interactions between different hosts either inside or outside a network system, thus providing abundant information for detecting intrusions. In previous papers we have introduced the design and implementation of VisFlowConnect-IP [9, 10, 11, 12], an animated tool for visualizing network flows. This paper describes how to use that tool in detail.

5 Example Anomaly Detection

Here, we show an example of how we can detect the blaster virus with VisFlowConnect-IP. The blaster virus spreads quickly and has a common worm characteristic in which infected computers send out packets to an abnormally large number of hosts within a short time period. In Figure 7, one can see that there is one domain which connects to almost every host in the local network. This indicates that some hosts in that domain might be infected by a worm. This is verified when we see the uniform payload size and port usage on all of these flows that match the Blaster signature. At this point we can filter on those characteristics, and by digging deeper with the domain view, we can begin to identify specific hosts that have been infected.

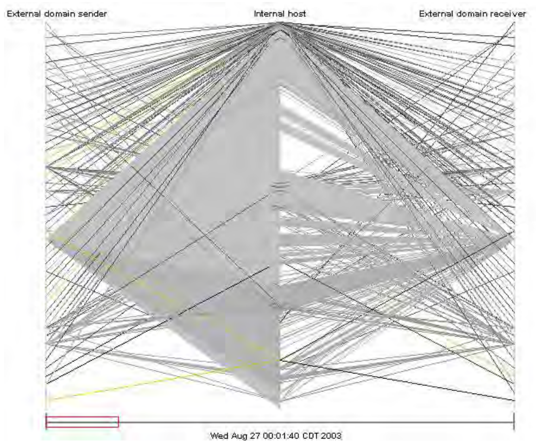


Figure 7. External view of blaster attacks

6 Conclusions

We have presented VisFlowConnect-IP, an approach to visualizing patterns on a network with NetFlow log data. Its purpose is to enhance an administrator's situational awareness by providing an easy-to-use, intuitive view of NetFlow data using link analysis. The central aspect of this interface is the parallel axes view, used to represent the origin and destination of network traffic. A high-level overview of the data is provided first, and the user is provided the capability of drilling down into the data to find additional details. Filtering mechanisms are provided in order to assist the user in extracting interesting or important traffic patterns. The VisFlowConnect visualization framework described in this paper is extensible beyond IP networks, and we are currently modifying it to monitor security in storage systems and high performance cluster computing environments as well.

References

[1] S. Axelsson. Visualisation for Intrusion Detection - Hooking the Worm. *Eighth European Symposium on Research in Computer Security (ESORICS)*, Lecture Notes in Computer Science (LNCS) , Springer, 2003.

[2] R. Ball, G. A. Fink, C. North. Home-Centric Visualization of Network Traffic for Security Administration. *CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

[3] G. Conti, K. Abdullah. Passive Visual Fingerprinting of Network Attack Tools. *CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

[4] R. Erbacher, K. Walker, D. Frincke. Intrusion and Misuse Detection in Large-Scale Systems. *IEEE Comp. Graphics and Applications*, 22(1):38–48, 2002.

[5] R. Henning, K. Fox. The Network Vulnerability Tool (NVT) – A System Vulnerability Visualization Architecture. *NISSC*, 2000.

[6] S. T. Teoh et al. Elisha: a Visual-based Anomaly Detection System. *RAID*, 2002.

[7] S. T. Teoh, K. Ma, S. F. Wu. A Visual Exploration Process for the Analysis of Internet Routing Data. *IEEE Visualization*, 2003.

[8] S. T. Teoh, K. Zhang, S. Tseng, K. Ma, S. F. Wu. Combining Visual and Automated Data Mining for Near-Real-Time Anomaly Detection and Analysis in BGP. *CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

[9] X. Yin, W. Yurcik, Y. Li, K. Lakkaraju, C. Abad. VisFlowConnect: Providing Security Situational Awareness by Visualizing Network Traffic Flows. *23rd IEEE Int'l. Performance Computing and Communications Conference (IPCCC)*, 2004.

[10] X. Yin, W. Yurcik, A. Slagell. The Design of VisFlowConnect-IP: a Link Analysis System for IP Security Situational Awareness. *3rd IEEE Int'l. Workshop on Information Assurance (IWIA)*, 2005.

[11] X. Yin, W. Yurcik, M. Treaster, Y. Li, K. Lakkaraju. VisFlowConnect: NetFlow Visualization of Link Relationships for Security Situational Awareness. *CCS Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC)*, 2004.

[12] W. Yurcik. The Design of an Imaging Application for Computer Network Security Based on Visual Information Processing. *SPIE Defense and Security Symposium / Visual Information Processing XIII*, 2004.